HTML

XHTML - CSS - Scripts

3ème ÉDITION

LE GUIDE COMPLET

« Maîtrisez l'univers HTML de A à Z ! »

Micro Application

Fabrice Lemainque

Copyright © 2009 Micro Application
20-22, rue des Petits-Hôtels
75010 Paris

1ère Édition - Avril 2009

Auteur Fabrice LEMAINQUE

ISBN : 978-2-300-019586

MICRO APPLICATION
20-22, rue des Petits-Hôtels
75010 PARIS
Tél. : 01 53 34 20 20
Fax : 01 53 34 20 00
http://www.microapp.com

Support technique :
Également disponible sur
www.microapp.com

Retrouvez des informations sur cet ouvrage !

Rendez-vous sur le site Internet de Micro Application **www.microapp.com**. Dans le module de recherche, sur la page d'accueil du site, entrez la référence à 4 chiffres indiquée sur le présent livre.
Vous accédez directement à sa fiche produit.

RECHERCHE
1958
OK
Livre

Avant-propos

Destinée aussi bien aux débutants qu'aux utilisateurs initiés, la collection *Guide Complet* repose sur une méthode essentiellement pratique. Les explications, données dans un langage clair et précis, s'appuient sur de courts exemples. En fin de chaque chapitre, découvrez, en fonction du sujet, des exercices, une check-list ou une série de FAQ pour répondre à vos questions.

Vous trouverez dans cette collection les principaux thèmes de l'univers informatique : matériel, bureautique, programmation, nouvelles technologies...

Conventions typographiques

Afin de faciliter la compréhension des techniques décrites, nous avons adopté les conventions typographiques suivantes :

- **gras** : menu, commande, boîte de dialogue, bouton, onglet.
- *italique* : zone de texte, liste déroulante, case à cocher, bouton radio.
- `Police bâton` : Instruction, listing, adresse internet, texte à saisir.
- ✂ : indique un retour à la ligne volontaire dû aux contraintes de la mise en page.



Il s'agit d'informations supplémentaires relatives au sujet traité.



Met l'accent sur un point important, souvent d'ordre technique qu'il ne faut négliger à aucun prix.



Propose conseils et trucs pratiques.



Donne en quelques lignes la définition d'un terme technique ou d'une abréviation.

Chapitre 1 Introduction à HTML 11

1.1. Bref historique de HTML 12
1.2. Pendant ce temps 17
XML 17
CSS (Cascading Style Sheet) 18
DOM2 (Document Object Model level 2) 20
1.3. La nouvelle génération : XHTML 21
1.4. Présentation rapide de la syntaxe et de la structure du langage 22
1.5. Résumé 25

Chapitre 2 Les outils Web 27

2.1. Outils de création de pages Web 28
Éditeurs commerciaux 29
Éditeurs gratuits 31
2.2. Navigateurs 36
Bref historique : Netscape contre Microsoft 37
De nouveaux challengers 38
2.3. Résumé 42

Chapitre 3 Conception de votre première page Web 43

3.1. Conception du site 44
3.2. Création de la page d'accueil 47
Instruction !DOCTYPE 47
Élément HEAD 49
Élément BODY 53
Paragraphes de texte : élément P 57
Lien hypertexte : élément A 59
Élément saut de ligne BR 65
Attributs d'alignements 66
3.3. Résumé 69

Chapitre 4 Listes et tableaux 71

4.1. Listes 73
Liste ordonnée : éléments OL et LI 73

Liste à puces : éléments UL et LI ... 75
Liste de définitions : éléments DL, DT et DD ... 76
Listes imbriquées ... 79
4.2. Tableaux ... 89
Création d'un tableau simple ... 90
Contrôle de la taille et de la structure des cellules ... 98
Imbrication de tableaux ... 127
Autres précisions concernant les tableaux ... 128
4.3. Résumé ... 133

Chapitre 5 Mise en forme avancée 135

5.1. Principaux éléments structuraux ... 136
Élément HR ... 136
Commentaire HTML ... 137
Modification des marges d'un paragraphe ... 148
5.2. Éléments de structuration du texte ... 149
Autres éléments structuraux ... 155
Élément OBJECT ... 155
5.3. Caractères spéciaux et encodage de caractères ... 156
Encodages de caractères ... 157
Caractères spéciaux ... 159
Caractères non affichables ... 163
5.4. Modification de l'apparence du texte ... 164
5.5. Notation mathématique ... 169
En texte simple ... 169
À l'aide de graphismes ... 170
À l'aide de MathML ... 171
5.6. Résumé ... 173

Chapitre 6 Couleurs et images 175

6.1. Les couleurs ... 177
Modification du schéma de couleurs de la page ... 178
Modification de la couleur du texte dans un élément de page ... 183
Modification du schéma de couleurs d'un tableau ... 184
Précautions à prendre avec les couleurs ... 184
6.2. Les images ... 186
Image d'arrière-plan ... 187

Image insérée 190

6.3. **Images animées 198**

Insertion d'un gif animé 199

Création d'un gif animé 203

6.4. **Image cliquable 210**

Images cliquables côté client : éléments MAP et AREA 210

Images cliquables côté serveur 222

6.5. **Inclusion générique d'images : élément OBJECT 224**

Diminution du temps de téléchargement 234

6.6. **Résumé 240**

Chapitre 7 Jeux d'encadrement 243

7.1. **Jeu d'encadrement : élément FRAMESET 245**

7.2. **Contenu d'un jeu d'encadrement : élément FRAME 248**

Attributs essentiels de l'élément FRAME : name et src 257

Attributs de mise en forme de l'élément FRAME 258

Autres attributs de l'élément FRAME 259

7.3. **Détermination du cadre cible : attribut target 261**

Noms réservés 262

Établissement de la cible par défaut des liens 264

Sémantique de cible 264

7.4. **Ensembles de cadres imbriqués 265**

7.5. **Partage de données entre cadres 269**

7.6. **Contenu de remplacement : élément NOFRAMES 270**

7.7. **Cadres en ligne : élément IFRAME 271**

7.8. **Travail avec les jeux d'encadrement 273**

7.9. **Résumé 275**

Chapitre 8 Feuilles de style 277

8.1. **Introduction 279**

8.2. **Attribut style 281**

8.3. **Élément STYLE 282**

Définition du style de la totalité des occurrences d'un élément donné 283

Définition du style de certaines occurrences d'un élément donné 286

Définition du style d'une unique occurrence d'un élément donné 287

Éléments DIV et SPAN 288
Attribut media 291
Masquage du contenu de l'élément STYLE 292
8.4. **Feuille de style externe 294**
Élément LINK 303
Prise en compte du médium 305
8.5. **Propriétés de feuille de style 306**
Propriétés de modification de texte et de police 306
Représentation des éléments à l'aide d'un style 311
8.6. **Feuille de style en cascade 323**
Cascade et héritage 323
8.7. **Feuille de style auditive, pour un public particulier 328**
8.8. **Résumé 335**

Chapitre 9 Formulaires 337

9.1. **Constituants d'un formulaire 339**
Élément FORM 339
Types de commandes de formulaire 341
Élément INPUT 343
Élément BUTTON 350
Éléments SELECT, OPTGROUP et OPTION 352
Élément TEXTAREA 356
Élément LABEL 357
Ajout d'une structure à un formulaire :
éléments FIELDSET et LEGEND 360
9.2. **Formulaire et focus 362**
Navigation par tabulation 363
Touches d'accès rapide 366
9.3. **Commandes inactives et en lecture seule 368**
Commandes inactives 368
Commandes en lecture seule 369
9.4. **Soumission du formulaire 370**
Méthodes de soumission du formulaire 370
Commandes réussies 371
Traitement des données du formulaire 373
Types de contenu du formulaire 374
9.5. **Remarques à propos des formulaires 375**
9.6. **Résumé 376**

Chapitre 10 Scripts 379

10.1. Introduction aux scripts 381
10.2. Élément SCRIPT 383
Définition du type de script : déclaration META et attribut type de l'élément SCRIPT 386
Autres attributs de l'élément SCRIPT 387
Cas des agents utilisateurs qui ne gèrent pas les scripts ... 388
10.3. Événements intrinsèques 390
10.4. Travail avec les scripts 396
JavaScript 397
ECMAScript 412
VBScript 413
Traitement de formulaires à l'aide de scripts 415
Cookies 416
10.5. Applets 424
Élément OBJECT 426
Élément PARAM 429
Déclaration et instanciation des objets 431
10.6. Résumé 432

Chapitre 11 Publication sur le Web 435

11.1. Identification du public 436
11.2. Accessibilité 438
Accessibilité des tableaux 441
Accessibilité des jeux d'encadrement 453
Modules complémentaires (plug-in) 454
11.3. Adaptation à plusieurs langues 456
Attribut lang 457
Attribut dir 461
Élément BDO 463
Langues et jeux de caractères 465
11.4. Test du site 466
Réactions du public 471
11.5. Publication sur le Web 472
11.6. Suivi de la fréquentation de votre site 478
11.7. Résumé 483

Chapitre 12 Annexes 487

12.1. Jeu des entités de caractères Latin-1 488
12.2. Éléments et attributs HTML 4.01 491
Éléments HTML 4.01 492
Attributs HTML 4.01 504
12.3. Compatibilité XHTML/HTML 513
Rédaction correcte des documents 513
Remarques sur les éléments 515
Remarques sur les attributs 516
Exclusions SGML 518
Instructions de traitement 518
Modèle Objet de Document et XHTML 519
Feuilles de style imbriquées (CSS) et XHTML 519
DTD XHTML 520
12.4. DTD et XML 523
Déclaration d'un élément 524
Déclaration d'attributs 526
Déclaration de notations 527
Déclaration d'entités 528
12.5. Sélecteurs CSS 530
12.6. Ressources Web 533
Spécifications 533
Outils de création Web 533
Navigateurs 534
Images 534
Feuilles de style 535
Scripts 535
Modules complémentaires 537
Logiciels FTP 537

Chapitre 13 Glossaire 539

Chapitre 14 Index 559

Chapitre 1

Introduction à HTML

Bref historique de HTML 12
Pendant ce temps 17
La nouvelle génération : XHTML 21
Présentation rapide de la syntaxe et de la structure du langage 22
Résumé 25

La lecture de cette introduction pourrait ne pas paraître à certains comme totalement indispensable à la compréhension du reste de l'ouvrage. Pourtant, connaître la genèse et l'évolution de HTML et des langages apparentés peut vous simplifier grandement la vie, en vous permettant de mieux comprendre les préoccupations et buts sous-jacents des créateurs et développeurs du langage. Pour la création d'un site Web comme pour n'importe quelle activité de développement, connaître l'historique et la philosophie du langage employé procure un avantage souvent déterminant pour une réussite parfaite.

1.1. Bref historique de HTML

Le langage HTML (*HyperText Markup Language*) a été développé initialement par Tim Berners-Lee, alors au CERN. Il a rapidement connu un vif succès grâce au navigateur Mosaic, développé au NCSA. Pendant les années 1990, il a poursuivi sa croissance en profitant de celle, explosive, du Web, et s'est enrichi de nombreuses manières. Le Web repose sur le respect, par les concepteurs de pages et les éditeurs de logiciels, de conventions identiques pour HTML, ce qui a motivé le travail commun sur les spécifications de HTML.

C'est un format non propriétaire fondé sur SGML (*Standard Generalized Markup Language*) se conformant au standard international ISO 8879. Il peut être créé et traité par de nombreux outils, depuis des éditeurs de texte simples jusqu'à des outils dédiés sophistiqués WYSIWYG (*What You See Is What You Get* ou tel écran, tel écrit). HTML emploie des balises (comme `<H1>` et `</H1>`) pour structurer un texte en en-têtes, paragraphes, listes, liens hypertextes, etc.

La spécification HTML 2.0 (RFC 1866 de novembre 1995) a vu le jour sous le contrôle de l'IETF (*Internet Engineering Task Force*). Le groupe de travail HTML du W3C (*World Wide Web Consortium*) diffuse en janvier 1997 la spécification HTML 3.2, qui apporte plusieurs améliorations et modifications.



États d'avancement d'un document W3C

Tout document W3C passe par plusieurs états d'avancement (ou de maturité) portant un nom précis.

Brouillon (WD, *Working Draft*) : un document publié par le W3C afin qu'il soit examiné par la communauté composée des membres du W3C, du public et des autres organismes techniques.



Candidat à la recommandation (CR, *Candidate Recommendation*) : le W3C considère que le document a été largement commenté et répond aux exigences techniques du groupe de travail (*Working Group*). Il publie un CR pour obtenir des commentaires sur les mises en œuvre.

Proposition de recommandation (PR, *Proposed Recommendation*) : c'est désormais un document technique mature, dont la possibilité de mise en œuvre a été largement vérifiée. Il est envoyé au Comité Consultatif (*Advisory Committee*) pour adoption finale.

Recommandation (REC, *Recommendation*) : une Recommandation W3C est une spécification ou un ensemble de règles ayant reçu l'approbation du W3C. W3C en recommande un large déploiement. Elle est analogue à un standard tel que diffusé par d'autres organismes.

Viendra ensuite la spécification HTML 4, un progrès immense par rapport aux versions antérieures. Ses principales innovations concernent l'internationalisation, l'accessibilité, les tableaux, les documents composés, les feuilles de style, les scripts et l'impression.

- **Internationalisation** : les documents peuvent être écrits en toutes les langues et diffusés partout dans le monde. Cela a été accompli en tenant compte du document RFC 2070, qui traite de l'internationalisation de HTML. L'adoption de la norme ISO/IEC:10646 comme jeu de caractères du document pour HTML a représenté une étape importante. C'est la norme mondiale la plus complète, qui tient compte des problèmes concernant la représentation des caractères internationaux, le sens des écritures, la ponctuation et autres particularités des langues mondiales. Cela permet une indexation des documents par les moteurs de recherche, une typographie de qualité, la synthèse de la parole à partir du texte, la césure, etc.
- **Accessibilité** : au fur et à mesure de la croissance de la communauté du Web et de la diversification des capacités et aptitudes de ses membres, il devient crucial que les technologies employées soient appropriées à leurs besoins spécifiques. Le langage HTML a été conçu pour rendre les pages Web plus accessibles à ceux qui présentent des déficiences physiques. Les développements de HTML 4 qui ont été inspirés par le souci de l'accessibilité comprennent :
  - une meilleure distinction de la structure et de la présentation du document, en encourageant pour cela l'utilisation des

feuilles de style au lieu des éléments et attributs de présentation HTML ;

- l'amélioration des formulaires, ce qui comprend l'ajout de touches d'accès rapide (*access keys*), la possibilité de regrouper sémantiquement les contrôles des formulaires et les options de l'élément `SELECT` ainsi que l'ajout des étiquettes actives (*active labels*) ;
- la possibilité de baliser la description textuelle d'un objet incorporé (avec l'élément `OBJECT`) ;
- un nouveau mécanisme d'images cliquables côté client (l'élément `MAP`), qui permet aux auteurs d'intégrer des liens sous forme de texte et d'images ;
- l'accompagnement obligatoire des images incluses avec l'élément `IMG` et des images cliquables incluses avec l'élément `AREA`, d'un texte de remplacement, ainsi que des descriptions longues des tableaux, images, cadres, etc. ;
- la gestion des attributs `title` et `lang` pour tous les éléments, ainsi que des éléments `ABBR` et `ACRONYM` ;
- un éventail élargi des médias cibles (tty, braille, etc.) à utiliser avec les feuilles de style ;
- l'amélioration des tableaux, en y incluant des légendes, des regroupements de colonnes et des mécanismes pour faciliter la restitution non visuelle ;

- **Tableaux** : le nouveau modèle de tableau est fondé sur le document RFC1942. Les auteurs disposent maintenant d'un plus grand contrôle sur leur structure et leur disposition (par exemple, les regroupements de colonnes). Ils peuvent spécifier la largeur des colonnes et permettre aux agents utilisateurs d'afficher les données de table progressivement, au fur et à mesure du chargement, plutôt que d'attendre le chargement entier du tableau.
- **Documents composés** : le langage HTML offre maintenant une structure standard pour l'incorporation d'objets média et d'applications génériques dans les documents HTML. L'élément `OBJECT` (de même que ses ancêtres, les éléments plus spécifiques `IMG` et `APPLET`) fournit le moyen d'inclure des images, des séquences vidéo, de l'audio, des mathématiques, des applications spécialisées et d'autres objets dans un document. Il permet aussi aux auteurs de spécifier une hiérarchie de restitutions de

remplacement aux agents utilisateurs qui ne gèrent pas certaines restitutions particulières.

- **Feuilles de style** : les feuilles de style simplifient le balisage HTML et soulagent grandement HTML des responsabilités de la présentation. Elles donnent aux auteurs comme aux utilisateurs le contrôle de la présentation des documents (informations sur les polices de caractères, alignement, couleurs, etc.). Les informations de styles peuvent être spécifiées pour un élément ponctuel ou pour des groupes d'éléments. Elles peuvent se trouver à l'intérieur du document HTML ou dans une feuille de style externe. Les mécanismes qui associent une feuille de style à un document sont indépendants du langage de feuille de style. Avant l'apparition des feuilles de style, les auteurs disposaient d'un contrôle limité sur la restitution des pages. HTML 3.2 comprenait un certain nombre d'attributs et d'éléments permettant un contrôle de l'alignement, de la taille de la police de caractères et de la couleur du texte. Les auteurs abusaient également de tables et d'images pour la mise en pages. Le temps relativement long nécessaire pour que les utilisateurs mettent à jour leurs navigateurs implique que ces usages vont perdurer encore pendant un certain temps. Cependant, puisque les feuilles de style offrent des moyens de présentation plus puissants, le W3C fera graduellement disparaître nombre d'éléments et d'attributs de présentation HTML. Les éléments et attributs concernés sont marqués comme « déconseillés ».
- **Scripts** : les scripts permettent aux auteurs de concevoir des pages Web dynamiques (par exemple, des « formulaires intelligents » qui réagissent au cours de leur remplissage par l'utilisateur) et d'employer HTML comme support d'applications en réseau. Les mécanismes fournis pour associer HTML et scripts sont indépendants du langage de script.
- **Impression** : les auteurs voudront parfois aider l'utilisateur dans l'impression d'autres documents, en sus du document courant. Lorsque des documents font partie d'un ensemble, on peut décrire leurs relations en utilisant l'élément HTML `LINK` ou encore en utilisant le cadre de description des ressources (RDF) du W3C.

Fondamentalement, HTML 4 sépare bien plus efficacement la *structure* de la *présentation*. Les éléments et attributs de présentation HTML sont de plus en plus remplacés par d'autres mécanismes, en particulier les feuilles de style. Plus particulièrement, les anciens éléments `FONT` et `BASEFONT` sont désormais déconseillés.

La spécification HTML 4.01 est enfin une révision de HTML 4 qui corrige des erreurs et apporte certaines modifications à la version précédente. Vous pourrez trouver le texte en français de la spécification HTML 4.01 à l'adresse **www.la-grange.net/w3c/html4.01/cover.html** Cette spécification incorpore davantage d'emprunts au langage XHTML (*eXtensible HyperText Markup Language*), ce qui a permis d'alléger d'autant la spécification XHTML 1.0.

HTML 4.01 existe en trois « parfums ». Vous spécifiez la variante à employer en insérant une ligne au début du document. Chaque variante dispose de sa propre *définition de type de document*, ou DTD (*Document Type Definition*), qui spécifie les règles d'emploi de HTML de façon claire et succincte :

- Le DTD HTML 4.01 **strict** comprend tous les éléments et attributs qui ne sont pas déconseillés ou qui n'apparaissent pas dans les documents avec jeu d'encadrement. Pour les documents qui utilisent ce DTD, prendre la déclaration de type de document suivante :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

- Le DTD HTML 4.01 **transitoire** inclut la totalité du DTD strict auquel se rajoutent les éléments et attributs déconseillés (la plupart d'entre eux concernant la présentation visuelle). Pour les documents qui utilisent ce DTD, prendre la déclaration de type de document suivante :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

- Le DTD HTML 4.01 de **jeu d'encadrement** inclut la totalité du DTD transitoire complété des cadres (*frames*). Pour les documents qui utilisent ce DTD, la déclaration de type de document suivante est employée :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```



DTD

La définition de type de document, ou DTD (*Document Type Definition*), définit la structure d'un document, les éléments et attributs qui y sont autorisés, et le type de contenu ou d'attribut permis. Un document *bien formé* répond simplement aux exigences de la spécification, tandis qu'un document *valide* se conforme strictement aux règles établies par la DTD à laquelle il fait référence.



L'étude complète des DTD dépasse l'objectif de ce livre. Une excellente présentation est disponible à l'adresse http://developpeur.journaldunet.com/tutoriel/xml/031219xml_dtd1a.shtml

Dans l'ensemble de ce livre, nous nous conformerons à la spécification HTML 4.01.

1.2. Pendant ce temps...

Au cours de cette période, d'autres groupes de travail s'intéressaient à différents aspects plus ou moins intégrés progressivement dans les spécifications HTML successives.

XML

XML (*eXtensible Markup Language*) a été développé sous l'égide du W3C en 1996. Il décrit une classe d'objets nommés *documents XML* et décrit partiellement le comportement des programmes informatiques qui les traitent. XML est une application restreinte de SGML (*Standard Generalized Markup Language*, ISO 8879). Par construction, les documents XML sont des documents conformes SGML.

Ils sont constitués d'unités de stockage nommées « entités », renfermant des données analysées (*parsed*) ou non analysées (*unparsed*). Les données analysées sont composées de caractères, dont certains forment des données « caractère » et d'autres un balisage. Le balisage code une description de la structure logique de stockage du document. XML procure un mécanisme imposant certaines contraintes sur cette structure logique.

Un module logiciel nommé *processeur XML* sert à lire les documents XML et à procurer l'accès à leur structure et à leur contenu. Par définition, un processeur XML accomplit son travail pour le compte d'un autre module, l'application. Cette spécification décrit le comportement attendu d'un processeur XML : la façon dont il doit lire les données XML et les informations qu'il doit transmettre à l'application.

Les objectifs fondamentaux de XML sont les suivants :

- XML doit être largement utilisable sur Internet.
- XML doit prendre en charge une large gamme d'applications.
- XML doit être compatible avec SGML.
- L'écriture de programmes traitant les documents XML doit être facile.
- Les dispositifs facultatifs de XML doivent être limités voire, dans l'idéal, inexistants.
- Les documents XML doivent être clairs et lisibles par les humains.
- La conception XML doit pouvoir être effectuée rapidement.
- XML doit être formel et concis.
- Il doit être facile de créer des documents XML.
- Le côté abrupt du balisage XML importe peu.

Cette spécification, accompagnée de ses standards associés (Unicode et ISO/IEC 10646 pour les caractères, Internet RFC 3066 pour les balises d'identification de langue, ISO 639 pour les codes de noms de langue et ISO 3166 pour les codes de noms de pays), procure toutes les informations nécessaires pour comprendre XML Version 1.0 et construire les programmes informatiques qui le traitent.

CSS (Cascading Style Sheet)

Le langage CSS permet de définir des feuilles de style qui peuvent être appliquées à un site Web. Il permet la manipulation des styles appliqués à chaque balise HTML, *via* un langage de script.

La première Recommandation CSS (Recommandation W3C du 17 décembre 1996 révisée le 11 janvier 1999) a défini tout ce qui touche aux caractéristiques graphiques : couleurs, polices, tailles, etc.

Une seconde Recommandation (CSS2, 12 mai 1998) a été ajoutée pour gérer tous les problèmes de positionnement dynamique. Elle définit les feuilles de style en cascade, niveau 2. CSS2 est un langage de feuille de style qui permet aux auteurs et aux lecteurs de lier du style (comme les polices de caractères, l'espacement et un signal auditif) aux documents structurés (comme les documents HTML et applications XML). En

séparant la présentation du style du contenu des documents, CSS2 simplifie l'édition pour le Web et la maintenance d'un site.

CSS2 étant construit sur CSS1, toute feuille de style valide en CSS1 est ainsi également valide en CSS2, à quelques rares exceptions près. CSS2 prévoit des feuilles de style liées à un média spécifique, ce qui autorise les auteurs à présenter des documents sur-mesure pour les navigateurs visuels, les synthétiseurs de parole, les imprimantes, les lecteurs en braille, les appareils portatifs, etc. Cette spécification introduit aussi les notions de positionnement du contenu, de téléchargement des polices, de mise en forme des tables, de fonctions d'internationalisation, de compteurs et de numérotage automatique et quelques propriétés concernant l'interface utilisateur.

Aujourd'hui, un troisième projet (CSS3, Brouillon du 23 mai 2001) est en cours de réalisation. Il modularise CSS2. Plusieurs de ces modules sont aujourd'hui à l'état de Candidats à la recommandation. Pour de plus amples informations, consultez le site du W3C.



À propos du DHTML

Le DHTML (*Dynamic HyperText Markup Language*) a été inventé par Netscape à partir de sa version 4. Ce n'est pas à proprement parler un langage de balises pour Internet : il n'existe d'ailleurs aucune norme DHTML à part entière. C'est en réalité un ensemble de technologies Internet associées afin de fournir des pages HTML plus dynamiques. Microsoft a suivi cette voie en développant une autre version de DHTML à partir de la version 4 d'Internet Explorer.

DHTML s'appuie sur HTML (nécessaire pour présenter le document), sur les feuilles de style (CSS), qui permettent de définir un style pour plusieurs objets et le positionnement de ceux-ci sur la page, sur le Modèle Objet de Document (DOM), proposant une hiérarchie d'objets afin de faciliter leur manipulation, et enfin sur un langage de script, essentiel pour définir des événements utilisateur. Il s'agit essentiellement de JavaScript (développé par Netscape) ou de JScript (développé par Microsoft), et éventuellement de VBScript, la tendance étant toutefois d'employer désormais ECMAScript, une tentative de normalisation du noyau du langage : sa syntaxe, ses mots-clés et ses composants natifs. La troisième édition du standard ECMA-262 a été publiée en décembre 1999 (**www.ecma-international.org/publications/standards/Ecma-262.htm**).

Sans script, le DHTML n'est pas dynamique. C'est l'association avec le script qui permet d'apporter des modifications après le chargement de la page, chose impossible avec le HTML classique. Avec ce dernier, une fois la page chargée et affichée, il est impossible d'afficher de nouveaux éléments ou de les déplacer.



Le DHTML serait très intéressant à utiliser s'il existait une norme officielle respectée par les navigateurs, ce qui reste loin d'être le cas : un script écrit pour un navigateur risque fort de ne pas fonctionner sur un autre sans travail d'adaptation. Même si chacun peut potentiellement profiter du DHTML, puisque pratiquement plus aucun internaute n'utilise de navigateur de génération inférieure à la version 4, les nombreuses incompatibilités entre navigateurs provoquent de grandes difficultés.

DHTML n'est donc en rien comparable au PHP, à l'ASP, aux CGI, qui permettent de formater « à la volée » les pages d'un site (souvent interfacé avec des bases de données) en proposant du contenu en temps réel et en interagissant avec l'utilisateur. La majorité des effets du DHTML restent ainsi réservés aux intranets, où la population des navigateurs est connue et maîtrisée.

DOM2 (Document Object Model level 2)

Le Modèle Objet de Document (DOM, Recommandation W3C du 13 novembre 2000) est une interface de programmation d'application (API) pour des documents HTML valides et XML bien formés. Il définit la structure logique d'un document et la manière d'y accéder et de le manipuler. Dans la spécification DOM, le terme « document » est employé au sens large : XML sert de plus en plus à représenter de nombreuses sortes d'informations, qui peuvent être stockées sur des systèmes variés, et étaient traditionnellement considérées comme des données plutôt que comme des documents. XML présente néanmoins ces données comme des documents, le DOM permettant de gérer ces données.

Avec le Modèle Objet de Document, les programmeurs peuvent construire des documents, naviguer dans leur structure ainsi qu'ajouter, modifier ou effacer des éléments et leur contenu. Tout ce qui se trouve dans un document HTML, ou XML, peut être touché, modifié, effacé ou ajouté en utilisant le Modèle Objet de Document, à quelques rares exceptions près.

Un objectif important du DOM en tant que spécification W3C est de fournir une interface de programmation standard pouvant être utilisée dans une grande diversité d'environnements et d'applications. Le DOM est conçu pour une utilisation dans n'importe quel langage de programmation.

1.3. La nouvelle génération : XHTML

XHTML (*eXtensible HyperText Markup Language*) constitue une famille de modules et de types de documents existants ou futurs, qui constituent une reproduction, un sous-ensemble et une extension de HTML reformulé en XML. Les types de documents de la famille XHTML sont tous fondés sur XML dans le but de fonctionner avec des agents utilisateurs fondés eux aussi sur XML. C'est le successeur de HTML, disposant désormais de sa propre spécification, XHTML 1.0. Celle-ci est la première Recommandation W3C pour XHTML, créée d'après les travaux antérieurs sur HTML 4.01, HTML 4.0, HTML 3.2 et HTML 2.0. Elle combine la force de HTML 4 avec la puissance de XML.

XHTML 1.0 emprunte les éléments et attributs de la Recommandation W3C HTML 4, ce qui permet son interprétation par les navigateurs existants tant que certaines règles très simples sont respectées (voir Annexe C). Vous pouvez donc employer dès à présent XHTML.

Vous pouvez transformer vos anciens documents HTML en XHTML grâce à un convertisseur d'*Open Source*, HTML Tidy. Cet outil élimine au passage certaines erreurs de balisage, réduit l'éparpillement et améliore globalement le balisage, facilitant d'autant sa maintenance ultérieure.

Comme HTML 4.01, XHTML 1.0 existe en trois « parfums » : *Strict*, *Transitional* et *Frameset*. Vous choisissez la variante en insérant une ligne au début du document, spécifiant le DTD employé.

La spécification XHTML 1.0 complète en anglais est disponible en plusieurs formats, dont HTML, PostScript et PDF. Il existe également de nombreuses traductions, dont une version française disponible à l'adresse **www.la-grange.net/w3c/xhtml1/**

Nous signalerons lorsque nécessaire les modifications à apporter au code HTML 4.01 pour une parfaite compatibilité avec la spécification XHTML 1.0.

1.4. Présentation rapide de la syntaxe et de la structure du langage

HTML est fondamentalement un langage très simple, dans lequel des *balises* (*tag*) définissent un élément et ses propriétés. En HTML, ces balises sont les signes inférieur et supérieur (comme dans `<TITLE>`). Une balise définit un *élément* : un objet, texte ou autre, encadré par une balise d'ouverture et une balise de fermeture. Cette dernière est identique à la balise d'ouverture, à ceci près qu'une barre oblique (*slash*) est placée entre l e caractère inférieur et le nom de l'élément. Entre les deux balises est placé le *contenu* :

```
<TITLE>Titre</TITLE>
```

Ici, l'élément est `TITLE`, la balise d'ouverture `<TITLE>`, le contenu `Titre` et la balise de fermeture `</TITLE>`.

Certains éléments peuvent exceptionnellement être présents dans un document sans balise : `<HEAD>` et `HEAD` sont tous les deux valides.

Les éléments dépourvus de contenu portent le nom d'*éléments vides*. Si cette absence de contenu est liée à la définition de l'élément, celui-ci est également dépourvu de balise de fermeture : un tel élément accomplit quelque chose, puis s'arrête, devenant superflu dans le code. Un saut de ligne (`BR`) est un tel élément vide : une fois le saut de ligne effectué, il n'y a plus rien à faire. Vous rencontrerez par la suite d'autres éléments vides.



XHTML

XHTML impose que tout élément possède une balise de fermeture. Il existe toutefois une notation particulière signalant en XTML un élément vide :

```
<br />
```

ou

```
<img src="image.jpg" alt="image" />
```

Cette construction est à préférer à `<BR></BR>`, autorisée par XML, mais qui provoque des résultats inattendus avec certains agents utilisateurs.



Un élément n’est pas une balise !

Certaines personnes font référence aux éléments comme à des balises (par exemple, « la balise P »). Souvenez-vous que l’élément `P` est une chose et que la balise `<P>` en est une autre. Par exemple, l’élément `HEAD` est toujours présent, même si les balises ouvrante `<HEAD>` et fermante `</HEAD>` peuvent être absentes du document.

Les éléments se répartissent en deux catégories principales : les blocs et les lignes, qui équivalent réciproquement aux styles paragraphe et caractère d’un traitement de texte. Un élément de type bloc crée un nouveau bloc (texte ou graphique), générant le plus souvent un saut de ligne ou de paragraphe lors de son ouverture et/ou de sa fermeture. Le plus classique des éléments de type bloc est le paragraphe de texte, mais il en existe bien d’autres, comme vous le verrez.

En revanche, un élément ligne ne provoque pas de saut de ligne ou de paragraphe et n’affecte pas la totalité du bloc. C’est le cas des éléments modifiant la police, les couleurs, etc. Quelques très rares éléments peuvent être à la fois de type bloc et de type ligne.

Tout élément peut posséder des éléments enfants imbriqués. Il porte alors le nom d’élément parent :

```
<H1>Ceci est le titre <B>important</B></H1>
```

Ici, un élément `B` (gras) est imbriqué dans l’élément parent `H1` (en-tête de niveau 1). Le contenu de l’élément `H1` est `Ceci est le titre <B>important</B>`, celui de l’élément `B` est `important`.

Les éléments peuvent posséder des propriétés associées, appelées *attributs*, qui peuvent eux-mêmes posséder des valeurs (par défaut, ou encore définies par l’auteur ou par un script). Les couples attribut/valeur apparaissent avant le caractère > final de la balise ouvrante d’un élément. Un nombre quelconque de couples attribut/valeur (autorisés), séparés par des espaces, peuvent apparaître dans la balise ouvrante d’un élément. Ils peuvent survenir dans n’importe quel ordre :

```
<IMG src="image.gif">
```

Les attributs sont obligatoires pour certains éléments.



Sensibilité à la casse

Les éléments HTML et leurs attributs ne sont pas sensibles à la casse. Si théoriquement vous n'avez donc pas à vous préoccuper de la bonne utilisation de majuscules et de minuscules dans vos balises, sauf pour les noms de fichiers, d'après la spécification, les noms de balise doivent toujours être écrits en majuscules et les attributs en minuscules.

En général, mieux vaut toujours placer la valeur d'un attribut entre guillemets, même si HTML tolère leur absence lorsqu'il s'agit d'une valeur numérique (contrairement à XTML). Cette valeur placée entre guillemets doit être considérée comme sensible à la casse. Cette sensibilité à la casse des valeurs d'attributs peut provenir du navigateur (exceptionnellement) ou du serveur (très fréquemment). Ainsi, une telle faute de casse peut passer inaperçue localement, mais être flagrante une fois le site publié.



XHTML

En revanche, XHTML est sensible à la casse : la spécification recommande d'écrire tous les noms d'éléments et d'attributs en minuscules. Pour XML, `<li>` et `<LI>` sont des balises différentes. Un soin particulier doit en outre être apporté au respect de la casse des valeurs des attributs.

Le Tableau 1.1 montre la structure d'un élément, illustrée de quelques exemples. Une balise HTML comporte toujours les signes <>.

Tableau 1.1 : *Structure d'un élément HTML*

Composant	Description	Exemple
<	Balise ouvrante de la balise d'ouverture	<
ELEMENT	Nom de l'élément	`A`
Attribut	Propriété de cet élément. Parfois obligatoire pour certains éléments.	`href`
Valeur	Valeur de l'attribut, précédée d'un signe égal et normalement placée entre guillemets.	`="http://www.microapplication.fr"`
>	Balise fermante de la balise d'ouverture	>

Tableau 1.1 : *Structure d'un élément HTML*		
Composant	**Description**	**Exemple**
Contenu	Contenu de l'élément (pas pour tous les éléments).	`MicroApplication`
</ELEMENT>	Balise de fermeture (parfois facultative).	`</A>`

Toute page HTML est composée, après la ligne de déclaration du DTD, d'un élément racine `HTML` qui comprend deux éléments enfants :

- L'en-tête (`HEAD`) comprend des données non-affichées par le navigateur, servant à sa configuration ou présentes au bénéfice de l'auteur, du lecteur ou d'autres intervenants.
- Le corps (`BODY`) contient les données qui seront affichées, encadrées par des balises spécifiant leur structure.

Une page HTML se présente donc fondamentalement comme suit :

```
<!DOCTYPE ...>
<HTML>
<HEAD>
...contenu...
</HEAD>
<BODY>
   contenu...
</BODY>
</HTML>
```

Nous reviendrons bien sûr plus en détail par la suite sur cette structure.

1.5. Résumé

- HTML est un langage conçu par le CERN au début des années 1990. Fondé sur SGML (*Standard Generalized Markup Language), il a connu plusieurs évolutions avant de parvenir à la spécification actuelle HTML 4.01.*
- HTML sépare désormais efficacement la *structure* de la *présentation*. Les éléments et attributs de présentation HTML sont de plus en plus remplacés par d'autres mécanismes, en particulier les feuilles de style.
- XML (*eXtensible Markup Language*), CSS (*Cascading Style Sheet*) et DOM (*Document Object Model*) sont autant de technologies

apparentées désormais intégrées ou exploitées en tout ou partie à et par HTML.

- XHTML représente la future génération de HTML. La spécification actuelle est XHTML 1.0.
- HTML est un langage dans lequel des *balises* (*tag*) définissent un élément et ses propriétés. En HTML, ces balises sont les signes inférieur et supérieur (<>). Une balise définit un *élément* encadré par une balise d'ouverture et une balise de fermeture. Le contenu figure entre ces deux balises.
- Il existe deux catégories principales d'éléments : les blocs et les lignes. Un élément de type bloc crée un nouveau bloc, générant le plus souvent un saut de ligne ou de paragraphe lors de son ouverture et/ou de sa fermeture. En revanche, un élément ligne ne provoque pas de saut de ligne ou de paragraphe et n'affecte pas la totalité du bloc de texte.
- Les éléments peuvent posséder des propriétés associées, ou *attributs*, dotées de valeurs. Les couples attribut/valeur apparaissent avant le caractère > final de la balise ouvrante d'un élément. Les attributs sont obligatoires pour certains éléments.

Chapitre 2

Les outils Web

Outils de création de pages Web 28
Navigateurs 36
Résumé 42

Pour bien concevoir un site Web, vous devez connaître et maîtriser deux types d'outils : un ou plusieurs éditeurs HTML, qui permettent de créer effectivement les pages et donc le site Web, et un ou plusieurs navigateurs, qui affichent réellement ces pages Web.

Vous pourriez penser : « d'accord, parler des logiciels de création de sites est logique, mais pourquoi aborder les navigateurs ? » La réponse est simple : malgré l'existence des Recommandations du W3C, la « petite guerre » entre les grandes familles de navigateurs ainsi que leur obstination à préconiser plutôt telle ou telle méthodologie (et surtout pas celle utilisée par l'autre) fait que le rendu d'une même page HTML peut grandement différer selon le navigateur employé.

2.1. Outils de création de pages Web

Un éditeur HTML a pour but d'éviter la saisie manuelle souvent fastidieuse des caractères spéciaux, des codes couleur, de l'encodage des balises, etc. Comme il en existe un grand nombre, quelques précisions sur les outils disponibles peuvent être utiles. Remarquez toutefois que toute appréciation en ce domaine relève du subjectif : nous n'en présentons que quelques-uns, classés par ordre alphabétique. La plupart des éditeurs étant disponibles sur le Net en version d'évaluation, il peut être utile d'en tester plusieurs avant de procéder à un choix définitif.

Les différents outils peuvent être classés en deux catégories : les éditeurs commerciaux et les produits *Open Source* (gratuits). Ils se répartissent également entre éditeurs WYSIWYG (*What You See Is What You Get*) et les éditeurs en texte brut, qui nécessitent de connaître le HTML. Ces derniers disposent généralement d'une fonction de prévisualisation dans laquelle le codage des balises HTML n'apparaît plus. Gardez cependant toujours à l'esprit que cette prévisualisation n'est, et ne sera, jamais fidèle à 100 % par rapport à ce qui sera affiché par un navigateur.

À notre avis, mieux vaut éviter les éditeurs « qui écrivent tout pour vous ». Un éditeur doit vous faciliter le travail tout en vous laissant la main dans l'élaboration de votre page. Adoptez l'éditeur avec lequel vous vous sentez le plus à l'aise. Même s'il n'est pas le plus célèbre, il sera votre compagnon de longues heures de travail. D'ailleurs, pourquoi se limiter à un seul éditeur ? Il est tout à fait envisageable d'employer un éditeur évolué pour les premiers jets, afin de disposer rapidement d'un

brouillon exploitable, puis de recourir à un éditeur de la première génération (plus proche des balises brutes) pour une personnalisation plus pointue.

Éditeurs commerciaux

Adobe Golive CS2 (Windows, MacIntosh)

Adobe Golive est avant tout destiné aux infographistes. Il bénéficie du savoir-faire d'Adobe en matière d'images et d'illustrations et permet la gestion de la mise en page au pixel près, la gestion de l'interactivité grâce à une ligne du temps et à un éditeur JavaScript, la création des feuilles de style WYSIWYG et la gestion votre site. Les autres concepteurs se sentent un peu perdus dans cet univers de palettes et autres outils. Le code généré est d'une complexité certaine, à faire pâlir tout puriste du codage HTML, tandis que son prix reste élevé (479 €).

Dreamweaver MX 2004 (Windows, Mac)

Macromedia Dreamweaver est considéré par beaucoup comme le meilleur éditeur HTML du moment. Il permet de concevoir intelligemment la création de sites (codage HTML propre et académique, compatibilité avec les différents navigateurs, mises à jour dans tout le site, etc.). Certainement un peu déroutant lors de la première prise en main, en raison de son nombre impressionnant de palettes d'outils, il est facile de s'y habituer : cet éditeur WYSIWYG convient aussi bien aux débutants qu'aux initiés. Complet, performant, professionnel et évolutif, son seul handicap est son prix élevé (530 €, mais seulement 260 € en version Éducation).

FrontPage (Windows)

Les utilisateurs de Microsoft Office, et plus spécialement de Word, ne seront pas dépaysés par l'interface de l'éditeur de Microsoft : sa prise en main leur est ainsi aisée. C'est un bon produit, auquel on ne peut reprocher qu'une façon peu académique d'écrire le HTML : il est fréquent que des navigateurs autres qu'Internet Explorer éprouvent quelques difficultés à interpréter des pages écrites avec cet éditeur (240 €).

HotDog Professional 7.0.1 (Windows)

La nouvelle version de la suite HotDog Professional Webmaster, de la société Sausage Software, prend en charge un nombre encore plus grand de langages (HMTL, JavaScript, CSS, PHP, ASP, VBScript). Particulièrement puissante, elle dispose d'à peu près toutes les fonctions envisageables. L'interface s'ouvre sur une fenêtre mixte qui présente à la fois le code et sa mise en page obtenue *via* le navigateur. De très nombreux outils et fonctions sont disponibles. Les outils Supertoolz peuvent être téléchargés en groupe ou séparément depuis le site Internet de Sausage Software. L'utilisation de la couleur en fonction des balises ou de la syntaxe, la gestion de sites, la prise en charge des codes de classement PICS (filtrage des contenus indésirables) et des canaux Web d'envoi de données à l'instigation du serveur sont quelques-unes des possibilités offertes. HotDog affiche automatiquement en rouge toute erreur de syntaxe HTML qu'il détecte. Cette version intègre Xara MenuMaker (pour la création de menus animés en DHTML) ainsi que Bloomer (facilite la création d'animations flash) (100 $).

Namo WebEditor

Intermédiaire entre FrontPage Express et FrontPage 2000, cet éditeur devrait ravir les webmestres débutants à intermédiaires refusant de recourir à des « usines à gaz » plus performantes, mais aussi plus complexes. Raisonnablement complet et intelligemment conçu, le prix de Namo WebEditor reste raisonnable (99 $).

WebExpert 6.0 (Windows)

WebExpert est un des rares éditeurs de ce groupe à ne pas être WYSIWYG, mais à proposer un volet de visualisation permettant de vérifier instantanément le résultat de son codage. Intéressant pour ceux qui préfèrent avoir la sensation de coder du HTML plutôt que celle d'employer un traitement de texte. C'est un éditeur complet, intelligent, respectueux du code et de l'esprit HTML. Disposant de scripts et d'aides intégrés, il permet de travailler rapidement et efficacement sur (X)HTML, CSS, JavaScript, ASP et PHP. Il possède un autre intérêt : son faible prix (67 €). Vous pouvez facilement en trouver une version d'évaluation de 30 jours en téléchargement.

Web Fast (Windows)

Web Fast est plus un outil complet de gestion de site qu'un simple éditeur Web. Dès son lancement du logiciel, il prend en charge les étapes nécessaires à la construction du site. Web Fast intègre le site lui-même, le logiciel FTP nécessaire à sa publication sur Internet, les fonctions de commerce électronique (versions Plus et Pro) avec suivi de commandes, le paiement sécurisé (version Pro), les statistiques complètes du site et un back-office. Le prix s'échelonne selon la version de 75 à 456 €.

Il existe bien d'autres éditeurs commerciaux, parmi lesquels HotMetal (500 $, voir www.xmetal.com/) et CoffeeCup HTML (www.coffeecup.com/html-editor/, 49 $). Vous pourrez les découvrir en lançant une recherche à l'aide de votre moteur de recherche favori, en employant comme mot-clé « html editor » ou « éditeur html ». Ici, comme souvent, Google reste votre meilleur ami !

Word (Windows)

Word peut également être considéré comme un éditeur HTML : il est possible d'enregistrer un texte au format HTML, le traitement de texte se chargeant d'ajouter toutes les balises nécessaires en fonction de ce qui a été saisi. Cela est malheureusement accompagné d'un fatras de balises absolument superflues, le respect des spécifications W3C restant interprété de façon très personnelle par Microsoft… À déconseiller absolument.

Éditeurs gratuits

Est-il réellement nécessaire de recourir à un logiciel payant ? Aucunement. Il existe de très nombreux produits gratuits, capables d'effectuer un excellent travail et probablement très suffisants pour la création d'un site personnel.

1st Page 2000 (Windows)

1st Page 2000 est un outil de création de sites Web aussi adapté à la création des pages HTML qu'à l'élaboration de scripts dans les langages de programmation les plus récents. Son interface disponible sous trois formes le destine tant aux débutants qu'aux professionnels du secteur.

Ce logiciel offre toutes les fonctionnalités courantes de ses concurrents payants, jusqu'aux plus sophistiquées comme en témoigne la présence d'un convertisseur XML, d'un vérificateur de syntaxe, d'un compresseur de documents ou encore d'un client FTP intégré. Il offre par ailleurs des possibilités originales comme le redimensionnement automatique des images, la correction de l'orthographe en cours de frappe (en anglais) et la prévisualisation dans quatre navigateurs différents. Outre l'édition en JavaScript particulièrement bien prise en charge, 1st Page 2000 permet de manipuler les fichiers Flash et Shockwave. Il sait identifier et surligner la syntaxe des langages DHTML, ASP, SSI, CFML et WebTV. Le paquetage comprend 450 scripts JavaScript, 15 en DHTML , 17 en Perl , 6 en HTML et 2 scripts CGI. Un bon niveau HTML est nécessaire. Il n'est actuellement disponible qu'en anglais et peut être trouvé en téléchargement un peu partout. Le site officiel se trouve à l'adresse **www.evrsoft.com/1stpage2.shtml**.

Amaya (Windows, Unix, AIX, Solaris, Linux)

Navigateur et éditeur WYSIWYG gratuit du W3C. Il sert à démontrer et à expérimenter la plupart des nouveaux formats et protocoles développés par le consortium W3. Il est adaptable et extensible, et prend en charge notamment MathLM. Amaya propose par défaut de créer un nouveau document uniquement dans une des trois versions de la spécification XHTML 1.0 (strict, transitoire, de jeu d'encadrement), mais il est possible de transformer le document en version HTML 4.01 (ou même en version XHTML 1.1) à l'aide de l'option de menu *File > Change the document type*. C'est un produit intéressant, ne serait-ce que pour vérifier la conformité de votre code (**www.w3.org/Amaya/**).

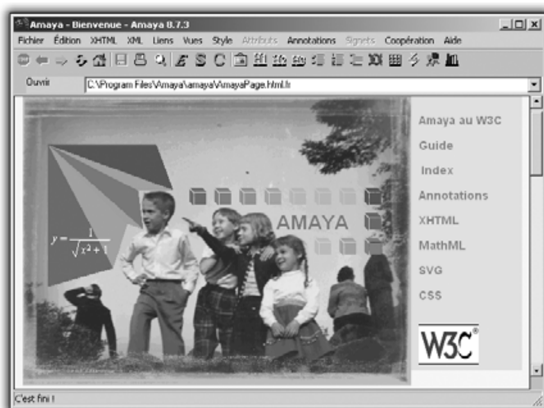


Figure 2.1 :
Interface d'Amaya

Éditeur Mozilla/Netscape Composer (Linux, Windows et Mac OSX)

Inclus dans la suite Netscape Communicator depuis la version 4.0, Netscape Composer était un éditeur gratuit. La dernière version de la suite Mozilla (actuellement la 1.7.7) contient toujours cet excellent éditeur, désormais nommé Mozilla Composer. Téléchargez-la en français sur **www.mozilla-europe.org/fr/** Vous disposerez, en plus du célèbre navigateur de l'éditeur, du logiciel gérant la messagerie et les forums et de celui de discussion en direct (Chatzilla).

L'éditeur Mozilla propose toutes les grandes fonctions d'un éditeur Web, dont des modes en code source ou en WYSIWYG, l'intégration de feuilles de style et un logiciel ftp (*file transfert protocol*) pour envoyer vos pages sur un serveur distant. Bien conçu, cet éditeur WYSIWYG reprend les principales, et donc les plus usuelles, fonctions du HTML. Il reste cependant un peu « léger » face aux éditeurs HTML plus spécialisés.

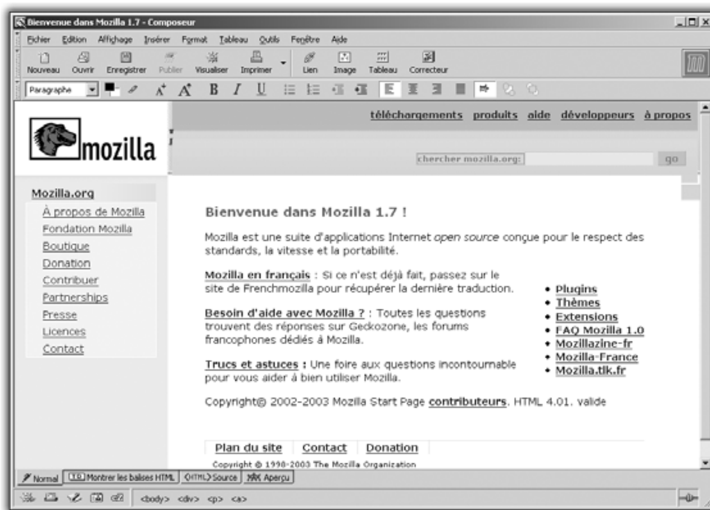


Figure 2.2 : *Éditeur Mozilla*

FrontPage Express

Inclus dans la suite Microsoft Internet Explorer 4.0, 5.0 et suivantes, FrontPage Express est également un éditeur gratuit. Il reprend les fonctions les plus usuelles du HTML. Son interface est voisine de celle

de Microsoft Word. Attention cependant, il « écrit » principalement pour Microsoft Explorer, ce qui peut provoquer de mauvaises surprises avec d'autres navigateurs. Prudence…

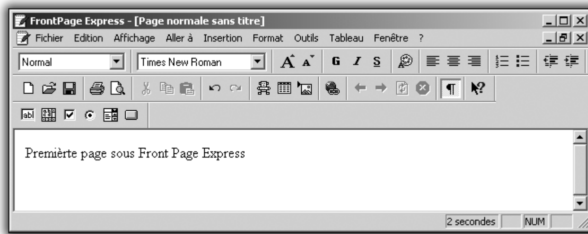


Figure 2.3 : *Front Page Express*

NotePad (Windows)

Eh oui, le bon vieux Bloc-Notes de Windows peut servir d'éditeur HTML ! Si vous employez Internet Explorer ou que vous en disposez, il est même probable que vous vous en servirez (ou que vous vous en êtes déjà servi) à votre insu : lorsque vous choisissez d'afficher le code source d'une page avec ce navigateur (**Affichage** >> **Source**), celui-ci s'affiche dans le Bloc-Notes. Si vous travaillez en local, vous pouvez immédiatement modifier le code, enregistrer le fichier, puis rafraîchir votre navigateur (bouton **Actualiser** ou **Affichage** >> **Actualiser**) pour voir les résultats de vos modifications ; et cela d'une façon souvent plus fidèle que celle du volet de prévisualisation des éditeurs plus évolués. Oserai-je avouer que, comme c'est le cas pour d'autres anciens programmeurs, le Bloc-notes reste mon principal éditeur HTML, surtout en raison du mécanisme qui vient d'être expliqué : j'ai pris l'habitude de travailler ainsi un site en local sous IE, même si mon navigateur Web principal est en réalité FireFox…

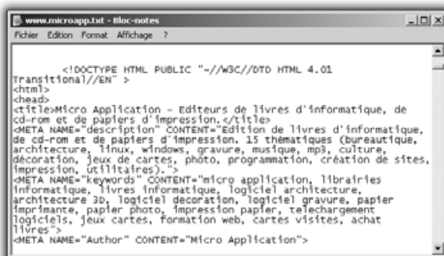


Figure 2.4 : *Affichage du code source dans le Bloc-Notes, depuis Internet Explorer.*

Nvu 1.0/KompoZer (Linux, Windows et Mac OSX)

Nvu (prononcer « N-viou ») est la refonte du module d'édition HTML de Mozilla développé par Disruptive Innovations. Il permet de créer facilement des pages Web en mode WYSIWYG. S'il n'est pas aussi puissant que les poids lourds comme Dreamweaver ou Adobe Golive, il reste largement suffisant pour la majorité des utilisations. Nvu permet de créer des pages Web et de gérer un site sans aucune connaissance technique ou apprentissage du HTML, ce qui le rend accessible à quasiment tout le monde. Il permet également de générer des documents en XHTML, d'éditer des pages en PHP et contient un éditeur de feuilles de style intégré (CaScadeS) très pratique. Vous trouverez plus d'informations sur Nvu sur le site officiel du projet Nvu (en anglais), **www.nvu.com**. Cerise sur le gâteau, il existe une version en français ! Produit recommandé.

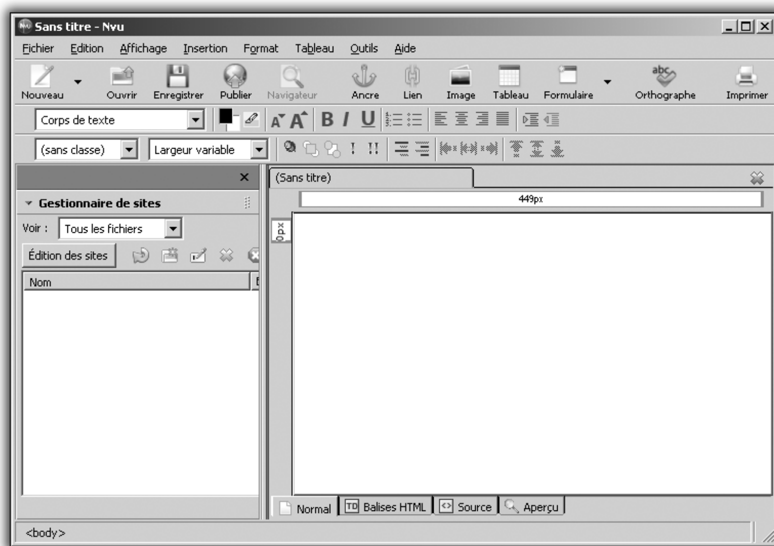


Figure 2.5 : *Interface de Nvu*

Nvu n'est toutefois plus maintenu et French Mozilla recommande le passage à Komposer (**http://www.kompozer.net/**). La version proposée en téléchargement est anglaise, mais il existe des kits de localisation disponibles sur le site officiel ou sur le site de French Mozilla (**http://frenchmozilla.sourceforge.net/nvu/**).

SoThink HTML Editor (Windows)

SoThink HTML Editor (anciennement nommé CutePage : vous pourrez encore le trouver sous ce nom) est un éditeur HTML doté de nombreuses fonctions, ainsi que d'un composant FTP capable de télécharger les fichiers sur un serveur. Des outils supplémentaires peuvent être téléchargés depuis le site de SourceTec Software. Appelés « widgets » (un *widget*, contraction de *Windows* et de *gadget*, désigne sous Unix et Linux un outil graphique permettant d'accomplir une action. Windows emploie plutôt le terme de « contrôle »), ils concernent en particulier la création de boutons, bannières et menus. SoThink est un éditeur qui apportera beaucoup aux débutants et auteurs de pages Web moyennement expérimentés (**www.sothink.com/htmleditor/**).

Comme pour les éditeurs commerciaux, il existe bien d'autres éditeurs gratuits. Là encore, vous pourrez en identifier en lançant une recherche à l'aide de votre moteur de recherche favori.

En conclusion, notre avis est qu'un logiciel gratuit est largement suffisant pour les besoins d'un utilisateur « normal » et peut suffire pour une petite ou moyenne entreprise. Comme nous l'avons déjà évoqué, le choix de ce logiciel dépend de vos préférences personnelles. Dans les exemples de ce livre, nous nous servirons d'un éditeur en texte brut (ici le Bloc-Notes de Windows, mais ce peut être un tout autre produit, par exemple Vi sous Linux), ainsi que des éditeurs Amaya du W3C et de l'excellent Nvu en français.

2.2. Navigateurs

Si vous souhaitez que votre site soit visible par tous dans de bonnes conditions, et qu'il ne soit pas réservé aux seuls francophones possédant une vue de 10/10, et disposant de surcroît exclusivement d'Internet Explorer version 6 corrigé SP2… vous devez prendre en compte les navigateurs de votre public. S'il s'agit d'un intranet d'entreprise, la chose est généralement facile. Si, en revanche, vous voulez vous ouvrir au monde entier, la tâche se complique. La première étape consiste donc à examiner les navigateurs actuellement employés et à connaître les principales différences existant entre eux.



Navigateur ou explorateur ?

Dans les premiers temps, Netscape Navigator était le seul produit existant. Le terme anglais *Web browser* a donc logiquement été traduit « navigateur Web ». Suite à l'arrivée d'Internet Explorer, Microsoft a insidieusement tenté de faire passer dans les mœurs le terme « explorateur Web ». Certains technocrates français ont alors tenté une solution intermédiaire, en proposant comme terme officiel « fureteur », tandis que les Canadiens préféraient « butineur ». Nous nous en tiendrons dans ce livre au terme consacré par l'histoire, soit « navigateur », malgré une tendresse personnelle pour butineur.

Bref historique : Netscape contre Microsoft

Comme nous l'avons déjà évoqué, le Web est né vers 1990 des travaux de Tim Berners-Lee. Le premier navigateur digne de ce nom, Mosaic, est créé en 1992 par le NCSA (*National Center for Supercomputing Applications*), un département de l'université de l'Illinois. Le projet était dirigé par Marc Andreessen. Ses études à peine terminées, ce dernier se rend dans la Silicon Valley, où il participe à la création de l'entreprise Mosaic Communications. Le nom Mosaic étant la propriété du NCSA, l'entreprise est rebaptisée Netscape.

Au milieu de l'année 1995, le navigateur de Netscape règne en maître : il était alors dépourvu de tout concurrent sérieux. Sa part de marché n'a pas été mesurée avec précision à cette époque, mais tout le monde s'accorde à dire qu'elle dépassait 80 %. Ayant pris conscience de l'importance du Web, Microsoft conçoit alors son propre navigateur, Internet Explorer, dont la première version est distribuée gratuitement avec Windows 95.

Dès 1996, la part de marché du navigateur de Netscape ne cesse de décroître, au rythme de 1 % par mois environ. Même si, comme l'a constaté la justice américaine, Microsoft a abusé de sa situation de monopole dans le domaine du système d'exploitation pour imposer son navigateur aux constructeurs de micro-ordinateurs, force est toutefois de reconnaître qu'à partir de la version 4, Internet Explorer l'emporte sur le plan technique : cette avance ne s'est ensuite jamais démentie.

En 1998, le code du navigateur Netscape est rendu public. Les logiciels libres et Netscape ont en effet un ennemi commun : Microsoft. En donnant à des bénévoles la possibilité de participer à l'évolution de son navigateur, Netscape pense probablement pouvoir accélérer et améliorer

sa production. C'est tout le contraire qui se passe. La version 5 du navigateur ne verra jamais le jour, et la version 6 fut, à ses débuts, très décevante (oserai-je dire que les termes « usine à gaz » et « lenteur extrême » sont ici des euphémismes ?) et surtout très tardive. Dès 1999, les parts de marché de Netscape et Microsoft sont respectivement de 29 et 69 % (source : Statmarket).

Un des gros défauts du navigateur de Netscape concernait l'affichage des tableaux. Lorsque ceux-ci commencent à être employés massivement pour la conception des pages Web, ce défaut éclate au grand jour. La version 6 est corrigée, malheureusement un peu tard : au début de l'année 2000, la part mondiale de marché du navigateur de Netscape n'est plus que de 18 %. Au milieu de l'année 2001, il était généralement admis qu'elle était descendue autour de 12 %, mais qu'elle était stabilisée. L'apparition de la version 6 du navigateur de Microsoft (fin octobre 2001) provoque une nouvelle baisse de la part de marché de son concurrent : elle descend en dessous de 10 % et continue à décroître, tandis que la version 6 d'Internet Explorer connaît en revanche un réel succès. La chute s'accélère alors de manière exponentielle. Une étude sur l'année 2001 créditait Navigator de 4,5 % de parts de marché, contre 94,2 % à Explorer.

La version 6.2 de Navigator, corrigée, apparaît enfin comme un produit sérieux, capable de rivaliser avec la version 6 de son concurrent, sauf en ce qui concerne la vitesse, pourtant améliorée. Netscape a misé sur la différence : son produit respecte les standards du W3C. Netscape a abandonné les technologies propriétaires (JavaScript) au profit de technologies standardisées (ECMAScript), alors que Microsoft reste complètement en marge des Recommandations W3C. Le rachat de Netscape par AOL place toutefois le sort de Navigator entre les mains de ces derniers : ce fournisseur d'accès à Internet procure toujours à ses clients un navigateur qui n'est autre que IE « personnalisé ».

De nouveaux challengers

En juin 2002, d'après OneStat, une société spécialisée dans l'analyse de trafic, Internet Explorer dépasse la barre des 95 %. L'arrivée de Netscape Navigator 7.0 en version d'évaluation au mois de mai, suivie par celle de Mozilla 1.0 au mois de juin, stabilise cependant le déclin de Navigator. Le prochain abandon de IE par AOL devient de plus en plus probable : Compuserve, filiale d'AOL, utilise désormais un navigateur à

moteur Gecko. Le navigateur de Netscape aurait donc finalement une chance de survivre.

Début 2005, avec 90 % du marché selon le cabinet d'études spécialisé WebSideStory, Internet Explorer est toujours, de loin, le navigateur Internet le plus utilisé dans le monde, bien qu'ayant perdu un peu plus de cinq points depuis juin, principalement au bénéfice de Firefox, l'application de la fondation à but non lucratif Mozilla. Celui-ci s'adjuge près de 5 % du marché en janvier, contre 2,7 % en octobre 2004. Dans les mois qui suivent, Firefox continue sa progression, directement aux dépens d'Internet Explorer.

Au cours de l'été 2005, Netscape jette l'éponge : la suite Mozilla 8, dont le nom de code était Seamonkey, ne verra jamais le jour. L'équipe rejoint la Fondation Mozilla pour se consacrer au développement de Firefox et de Thunderbird, le logiciel de messagerie frère.

Entre 2005 et 2008, Firefox progresse de près de 13 points, grignotant sur Internet Explorer et sur les autres produits basés sur Mozilla (la suite et Netscape). On peut remarquer les audiences de Safari (réservé aux derniers Mac OS X) à un peu plus de 1,5 %. Opera est désormais marginal avec 0,1 %.

Tableau 2.1 : *Parts de marché des navigateurs Web entre janvier 2005 et janvier 2008 d'après Libstat, Médiamétrie-eStat et Journal du Net (http://solutions.journaldunet.com/dossiers/chiffres/navigateurs.shtml et http://www.libstat.com/pages/navigateur.htm)*

Navigateur	Pourcentage au 01/2005	Pourcentage au 01/2007	Pourcentage au 01/2008
Internet Explorer	90,7 %	85,3 %	73,4 %
Mozilla/Firefox	8,1 %	11,7 %	20,7 %
Safari	1,2 %	1,6 %	1,5 %
Opera	0,3 %	0,6 %	0,1 %
Autres	0,6 %	0,3 %	3,3 %

Ces chiffres dépendent toutefois largement du pays et du type d'utilisateur : selon certaines données, la part de marché de Firefox pourrait déjà dépasser 35 % en France chez les utilisateurs individuels (hors entreprises).

Il est également intéressant d'examiner les systèmes d'exploitation employés. Windows Vista atteint 13 %, Windows XP est crédité de 77 %. Les anciens Windows ne représentent plus que 4 % des internautes français. Mac et Linux progressent tout en restant modestes avec respectivement 3,5 et 0,3 % de ces internautes en janvier 2008.

Concrètement, cela signifie que dans l'immédiat vous devez tester vos pages avec Internet Explorer (de préférence versions 6 et 7) et Firefox pour toucher près de 96 % du public potentiel. Si possible, effectuez également un test avec Safari (sur une machine Mac), Netscape et Mozilla (sur une machine Windows) et enfin avec Konqueror sur une machine Linux.

Tableau 2.2 : *Comparaison entre navigateurs Web (source : http://kb.mozillazine.org/Intro_:_Comparison)*

Dispositif	**Mozilla Suite/Firefox**	**Camino**	**Opera** (http://opera.com/)	**Apple Safari** (http://apple.com/safari/)	**Microsoft Internet Explorer** (http://microsoft.com/windows/ie/)
Plates-formes prises en charge	Windows, Mac OS X, Linux, autres	Mac OS X	Windows Linux, Symbian (http://symbian.com/), Motorola (http://motorola.com/), autres	Mac OS X	Windows
Coût	Gratuit	Gratuit	39 $ (ou gratuit avec publicités)	Intégré	Intégré
Open Source	Oui (http://mozilla.org/MPL/)	Oui (http://mozilla.org/MPL/)	Non	Partiellement (http://developer.apple.com/darwin/projects/webcore/)	Non
Onglets de navigation	Oui	Oui	Oui	Oui	Avec un programme de tierce partie (comme MyIE2)
Blocage des popup	Oui	Oui	Oui	Oui	Oui
Gestionnaire de formulaires/mots de passe	Oui	Oui	Oui	Oui	Limité *via* modules complémentaires
Thèmes	Nombreux (http://texturizer.net/firefox/themes/)	Quelques-uns	Nombreux (http://my.opera.com/community/customize/)	Quelques-uns	*Via* une application séparée

Tableau 2.2 : *Comparaison entre navigateurs Web (source : http://kb.mozillazine.org/Intro_:_Comparison)*

Dispositif	Mozilla Suite/Firefox	Camino	Opera (http://opera.com/)	Apple Safari (http://apple.com/safari/)	Microsoft Internet Explorer (http://microsoft.com/windows/ie/)
Extensions	Nombreuses (http://update.mozilla.org/, (http://extensionroom.mozdev.org/) et (http://extensionsmirror.nl/)	Néant	Néant	Quelques-unes	Quelques-unes
Personna-lisation	Intensive (avec **about:config**)	?	Intensive (avec les fichiers *prefs* et *.ini*)	?	Partielle (à l'aide des préférences et des fichiers du registre et de stratégies)

Vous trouverez dans l'Annexe C les adresses où vous procurer les différents navigateurs. Il convient également de citer Lynx (http://lynx.browser.org/), un navigateur en texte seul largement employé par les personnes souffrant de handicaps visuels. Il est souvent employé en combinaison avec un convertisseur texte-parole.



***Figure 2.6 :** Lynx, navigateur en texte seul, affichant la page d'accueil de Google.*



Convertisseurs texte-parole

De nombreuses personnes handicapées, particulièrement les malvoyants, ont recours à des convertisseurs texte-parole. Il s'agit de programmes capables de lire à haute voix un document informatique. Ces convertisseurs emploient souvent une inflexion particulière ou un signal lorsqu'ils rencontrent un élément HTML particulier, si bien qu'il est déconseillé d'utiliser de façon incorrecte ces éléments si votre public est susceptible d'inclure des handicapés. Et, à moins que votre site ne soit pas présent sur Internet, il est probable qu'il sera visité par un



nombre certain de handicapés. Ces convertisseurs font partie des *agents utilisateurs* HTML.



Pourquoi « agent utilisateur » plutôt que « navigateur » ?

Même si les navigateurs sont probablement les principaux consommateurs de documents HTML et XHTML, d'autres programmes et systèmes lisent ces documents. Par exemple les convertisseurs texte-parole que nous venons d'évoquer ou les moteurs de recherche qui lisent les pages Web sans être des navigateurs. Le terme « agent utilisateur » sert à rappeler cette nuance.

Une recherche sur Google montre ainsi parfois certains résultats stipulant « Cette page utilise des cadres, mais votre navigateur ne les prend pas en charge », ce qui peut dissuader certains utilisateurs de cliquer sur ce lien. L'auteur de ce site a oublié qu'il existait autre chose que des navigateurs : il aurait dû placer un texte approprié dans un élément `noframes` (reportez-vous au Chapitre 7, traitant des jeux d'encadrement, pour plus de précisions à ce sujet) pour éviter ce problème.

2.3. Résumé

- Un éditeur HTML a pour but d'éviter la saisie manuelle souvent fastidieuse des caractères spéciaux, des codes couleur, de l'encodage des balises, etc.
- Les éditeurs HTML peuvent être commerciaux ou gratuits, WYSIWYG (*What You See Is What You Get*) ou en texte brut.
- Les navigateurs ne traitent pas tous le HTML de la même façon : pour bénéficier d'une audience maximale, il est important de connaître ceux employés par votre public et d'adapter votre code en conséquence.
- Le marché des navigateurs évolue avec le temps : Netscape Navigator, au début le seul navigateur existant, a été concurrencé peu à peu par Internet Explorer qui l'a finalement supplanté. Un nouveau venu, Firefox, s'implante de plus en plus.
- Un « agent utilisateur » est un programme capable de comprendre le langage HTML. Les navigateurs sont des agents utilisateurs visuels, mais il existe d'autres agents utilisateurs, comme les convertisseurs texte-parole (agents utilisateurs audio), les robots des moteurs de recherche, etc.

Chapitre 3

Conception de votre première page Web

Conception du site 44
Création de la page d'accueil 47
Résumé 69

Vous avez choisi un ou plusieurs éditeurs de pages Web, avez téléchargé et installé éventuellement quelques navigateurs complémentaires à fin de test : vous êtes prêt à passer à la conception de votre site.

Éléments étudiés dans ce chapitre :

- `!DOCTYPE`
- `HTML`
- `HEAD`, `TITLE`, `META`
- `BODY`, `Hn`, `P`, `A`, `BR`

3.1. Conception du site

Tout d'abord, prenez un peu de temps pour réfléchir, si possible avec une feuille de papier et un stylo. Pour bien concevoir votre site dès les premiers instants, vous devez en prévoir dès le début sa structure et son contenu. Il sera bien sûr toujours possible de procéder à des modifications par la suite (vous n'y manquerez d'ailleurs pas), mais le temps consacré à cette réflexion initiale sera largement amorti !

Un site peut être considéré comme une arborescence contenant les éléments suivants :

- Une page d'accueil, avec une barre de navigation.
- Une ou plusieurs pages secondaires, dont certaines peuvent également comporter une nouvelle barre de navigation : cela correspond à une nouvelle ramification de l'arborescence.
- Éventuellement des pages tertiaires, et ainsi de suite…

Le site le plus simple ne comporte qu'une page, tandis que les sites complexes peuvent posséder de très nombreuses ramifications. Dans ce dernier cas, le site propose souvent une carte du site, destinée à faciliter la navigation des utilisateurs (voir Figure 3.1).

Sur votre feuille de papier, commencez à tracer l'arborescence de votre site, en figurant chaque page par un rectangle (pas trop petit !). Dans chaque rectangle, donnez un titre à la page, puis notez un aperçu du contenu prévisionnel.

N'hésitez pas à répartir des informations trop abondantes en plusieurs pages. Les questions à se poser à ce stade sont les suivantes :



Figure 3.1 : *Page de plan du site du portail de l'administration française (www.service-public.fr/aide/plan.html)*

- Quel est le but de ce site ?
- L'information proposée est-elle utile ?
- Une même page ne comporte-t-elle pas trop d'informations ? Si oui, est-il possible de diviser logiquement ces informations ?
- La structure obtenue est-elle équilibrée et pertinente ?

Dans le cadre de l'exemple de ce livre, le site envisagé est présenté dans la figure suivante.

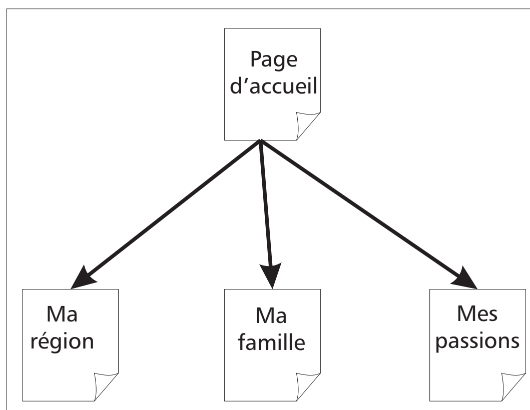


Figure 3.2 : *Ébauche du plan de notre site présenté dans l'ouvrage*

Cela fait, vous êtes prêt à débuter réellement la création de la première page de votre site Web. Vous allez commencer par la page la plus importante, la page d'accueil. À la fin de ce chapitre, vous aurez créé une page identique à celle de la figure qui suit, dont le code est présenté dans le listing suivant.



Figure 3.3 :
Page d'accueil telle que conçue à la fin de ce chapitre

Listing 3-1 : Page d'accueil telle que conçue à la fin de ce chapitre (version 1.3.3)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>"Ma page d'accueil"</TITLE>
<META name="author" content="mon nom">
<META name="version" content="1.3.3">
</HEAD>
<BODY>
<H1 align="center">Ma page d'accueil</H1>
<H2 align="center">Bienvenue sur mon site.</H2>
<P>Vous trouverez sur ce site des informations :<BR>
sur ma <A href="region.html">région</A> ;<BR>
sur ma <A href="famille.html">famille</A> ;<BR>
sur mes <A href="passions.html">passions</A>.</P>
<P align="center"><A href="mailto:votre_nom@votre_FAI">
Ecrivez-moi !</A>.</P>
</BODY>
</HTML>
```

Nous allons procéder pas à pas pour la création de cette page, en expliquant à chaque étape les instructions et éléments employés.

3.2. Création de la page d'accueil

La page d'accueil est la page la plus importante d'un site : c'est elle qui incitera le visiteur à examiner plus en détail le site ou à le quitter immédiatement. Il existe de nombreuses façons de rendre attrayante une page d'accueil (ou d'ailleurs toute page) : vous les découvrirez peu à peu au fil de la lecture de ce livre.

Il est fréquent de se contenter d'une ébauche de page d'accueil au moment de la construction du site, pour la fignoler en dernier lieu. Il ne s'agit donc pour le moment que de créer une première page relativement simple. Gardez bien présent à l'esprit que, à partir de HTML 4, le plus important concerne la structure de la page. La majorité des enjolivements sont facilement ajoutés par la suite.

La démarche est ici très similaire à celle recommandée lors de la rédaction d'un document à l'aide d'un traitement de texte : vous commencez par une frappe « au kilomètre », en privilégiant la structure (les phrases, paragraphes, chapitres et autres subdivisions) par rapport aux enrichissements qui ne seront apportés qu'ultérieurement, généralement à l'aide d'une feuille de style.

En effet, une page Web n'est en réalité guère plus qu'un fichier texte pourvu d'un ensemble de balises de spécification permettant l'affichage dans un navigateur Web (il en va de même dans votre traitement de texte, mais ces balises internes sont masquées).

Pour insister sur ce fait, les premiers travaux seront effectués uniquement à l'aide du Bloc-Notes. Nous vous conseillons de créer sur votre disque dur, à l'emplacement de votre choix, un nouveau dossier nommé par exemple *MonSiteWeb* : vous y rassemblerez tous les fichiers du site Web. Sinon, par défaut, le Bloc-notes devrait créer vos fichiers dans *Mes Documents*.

Instruction !DOCTYPE

HTML ayant connu de nombreuses révisions pendant son histoire, il était nécessaire d'informer l'*agent utilisateur* de la version HTML employée. Cela est effectué à l'aide de l'instruction `!DOCTYPE` (d'ailleurs automatiquement insérée en tête d'un document Web par la plupart des éditeurs HTML). Vous verrez par la suite comment utiliser

!DOCTYPE lorsque vous aurez recours à des feuilles de style en cascade, à des cadres ou à XML.

Commencez la création de la première page de votre site.

1 Ouvrez le Bloc-Notes (**Démarrer** >> **Tous les programmes** >> **Accessoires** >> **Bloc-notes Windows ou Démarrer** >> **Exécuter** >> **Notepad**).

2 Saisissez comme première ligne de votre document HTML :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

- L'instruction <!DOCTYPE> indique ici que ce document respecte la définition de type de document HTML 4.01, telle que définie par le W3C et disponible à l'adresse **www.w3.org/TR/REC-html4/loose.dtd** Nous choisissons ici le DTD transitoire pour pouvoir par la suite employer les éléments déconseillés (essentiellement ceux permettant de modifier la présentation dans le document HTML), interdits dans le DTD strict.
- Remarquez que cet élément est dépourvu de balise de fermeture. Vient ensuite l'élément HTML signalant le début d'un document HTML.

3 Appuyez sur [↵], puis saisissez <HTML> pour débuter la page. Appuyez sur [↵], puis fermez immédiatement l'élément HTML en saisissant sa balise de fermeture </HTML>.



Fermez toujours un élément après l'avoir ouvert

Même si HTML tolère un certain nombre d'erreurs de syntaxe, une bonne habitude à prendre consiste à toujours fermer un élément immédiatement après l'avoir ouvert, puis à saisir le contenu entre les deux balises. Cela évite d'oublier de fermer un élément par la suite, ce qui peut entraîner des résultats inattendus. C'est d'ailleurs ainsi que procède un éditeur HTML, comme vous le constaterez vous-même.

4 Par sécurité, enregistrez votre fichier sous le nom *pageacc1_0.html*. Assurez-vous de bien choisir une extension *.html*, même si *.htm* pourrait être suffisant, afin d'indiquer au navigateur que le fichier est de type HTML et doit être interprété en conséquence. Ne fermez ni le fichier ni le Bloc-notes.



Enregistrement au format HTML

Selon l’outil d’édition employé, enregistrer un document au format HTML peut être enfantin ou plus complexe. Un éditeur Web utilise généralement par défaut le format HTML (avec comme extension *.html* ou *.htm*). En revanche, un éditeur de texte simple, comme le Bloc-Notes, ou un traitement de texte, tel que Word ou WordPerfect, peuvent rendre nécessaire de modifier le type de fichier lors de l’enregistrement. Si vous éprouvez des difficultés à ouvrir un document Web créé à l’aide d’un traitement de texte ou d’un éditeur de texte simple, vérifiez l’extension du nom du fichier et le type de fichier, afin de vous assurer qu’il a bien été enregistré au format HTML.

Élément HEAD

Le premier élément enfant de l’élément `HTML` est l’élément `HEAD`. Celui-ci, l’en-tête du document, contient de nombreuses informations importantes qui intéressent le navigateur. À l’exception du contenu de l’élément `TITLE`, le contenu de l’élément `HEAD` n’apparaît pas dans ce qui est affiché directement dans le navigateur Web. Vous pouvez donc y placer des commentaires, des informations destinées aux moteurs de recherche, des avertissements relatifs aux droits d’auteur, ainsi que du code de script. C’est également dans cet élément que doivent se trouver les informations relatives aux feuilles de style. Cet élément est le premier du fichier HTML à être chargé et donc lu par le navigateur.

1. Revenez au document précédemment créé (*pageacc1_0.html*), s’il est toujours ouvert. Sinon, ouvrez le Bloc-Notes, puis ouvrez le fichier.
2. Placez-vous en dessous de la balise d’ouverture `<HTML>` (mais au-dessus de la balise de fermeture `</HTML>`), appuyez sur ↵, puis saisissez `<HEAD>` pour débuter la section d’en-tête. Le contenu de cet élément n’apparaît pas dans le navigateur Web. Comme vous l’avez fait précédemment, et pour les mêmes raisons, fermez immédiatement l’élément `HEAD` en appuyant sur ↵, puis en saisissant sa balise de fermeture `</HEAD>`.

Élément TITLE

Un élément indispensable dans l’élément `HEAD` est l’élément `TITLE` (titre), qui définit le texte affiché dans la barre de titre de la fenêtre de votre navigateur.

1 Placez-vous en dessous de la balise d'ouverture `<HEAD>`, appuyez à nouveau sur [↵], puis ajoutez la balise d'ouverture `<TITLE>` pour définir le titre de la page. Ce titre sera affiché dans la barre de titre de la fenêtre du navigateur. Fermez l'élément `TITLE` en saisissant `</TITLE>`.

2 Revenez juste après la balise `<TITLE>`, puis saisissez le titre de la page, ici `"Ma page d'accueil"`. Appuyez sur [↵].

Élément META

Autre composant capital de l'élément `HEAD`, l'élément `META` est un moyen de procurer des informations concernant votre site aux moteurs de recherche, à d'autres auteurs Web et à des serveurs Web. Vous pouvez y placer pratiquement n'importe quel type d'information. Cet élément est dépourvu de balise de fermeture.

Un élément `META` est un élément vide (dépourvu de contenu) dont la syntaxe est la suivante :

```
<META name="valeur" content="valeur">
```

L'attribut `name` définit ce que la valeur de `content` va décrire. Par exemple, la valeur de `name` peut être `"datetime"` : l'agent utilisateur sait alors que les chiffres et les lettres de `content` correspondent à une date et à une heure, probablement celles de la dernière modification. La balise de fermeture d'un élément `META` est facultative.



XHTML

Comme pour tout élément vide, un élément `META` doit être rédigé comme suit en XHTML :

```
<meta name="nom" content="contenu" />
```

Rien ne vous empêche d'adopter d'ores et déjà cette syntaxe dans vos documents HTML.

Un élément `HEAD` peut posséder un nombre quelconque d'éléments `META`. Ceux-ci permettent de fournir de nombreuses informations utiles concernant votre page, comme l'auteur ou la version, ce que vous allez faire maintenant.

1 Placez-vous sur la ligne vierge en dessous de l'élément `TITLE`. Saisissez `<META name="author" content="votre_nom">`

(où votre_nom est votre nom), appuyez sur [↵], puis saisissez <META name="version" content="1.0">.

2 Par sécurité, enregistrez à nouveau votre fichier sous le nom *pageacc1_0.html*. Ne fermez ni le fichier ni le Bloc-Notes.

Les éléments META sont en outre particulièrement utiles pour faire reconnaître votre page par les robots de recherche du genre AltaVista ou Google.

```
<META NAME="description" CONTENT="description de la page">
```

Cette balise indique que le contenu de CONTENT est la description de votre page HTML. Ce contenu pourra être affiché par un moteur de recherche comme résultat d’une recherche d’un utilisateur.

```
<META NAME="keywords" CONTENT="mot-clé1, mot-clé2, ...">
```

Cette balise indique que le contenu de CONTENT est une série de mots-clés qui définissent plus finement votre page.

Autre élément META fréquent, celui qui indique le nom du générateur du document, généralement inséré automatiquement par votre éditeur HTML :

```
<meta name="generator" content="nom_du_générateur">
```

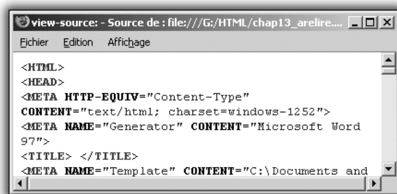


Figure 3.4 : *Certains éditeurs insèrent automatiquement diverses instructions META, dont le nom du générateur.*



Expressions anglophones

Rien n’impose que la valeur de name soit exprimée en anglais. Vous pouvez tout à fait légalement retenir name="auteur" et name="mots clés". Comme toutefois ces méta-informations sont généralement exploitées par des agents anglophones (la quasi-totalité des moteurs de recherche), mieux vaut, pour plus d’efficacité, conserver les expressions anglophones.

Toute personne examinant le code source de la page peut lire les informations présentes dans l’élément HEAD, mais celles-ci ne sont pas affichées dans le texte du document.



Affichage source

Pratiquement tous les navigateurs permettent d'examiner le code HTML source d'une page Web affichée dans votre navigateur, à l'aide d'une commande ressemblant à **Affichage** >> **Source**. Cette commande se trouve dans Internet Explorer, dans Netscape Navigator et dans Firefox dans le menu **Affichage** de la barre de menus.

L'élément `HEAD` doit également englober tout script et information de feuille de style, respectivement dans des éléments `SCRIPT`, `LINK` et `STYLE` (nous y reviendrons par la suite). Un exemple d'un élément `HEAD` simple est montré dans le Listing 3.2, correspondant à l'en-tête de la page présentée sur la figure suivante.



Figure 3.5 : Page d'accueil de MicroApplication

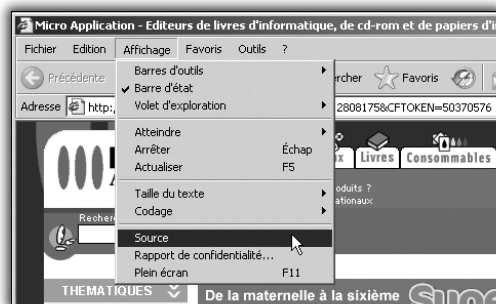


Figure 3.6 : Accès au code source de la page d'accueil de MicroApplication depuis Internet Explorer



Figure 3.7 : *Accès au code source de la page d'accueil de MicroApplication depuis Firefox*

Listing 3-2 : Extrait de la section HEAD de la page d'accueil de MicroApplication

```
<HEAD>
<TITLE>Micro Application - Editeurs de livres d'informatique,
de cd-rom et de papiers d'impression.</title>
<META NAME="description" CONTENT="Edition de livres
d'informatique, de cd-rom et de papiers d'impression. 15
thématiques(bureautique, architecture, linux, windows, gravure,
musique, mp3, culture, décoration, jeux de cartes, photo,
programmation, création de sites, impression, utilitaires).">
<META NAME="keywords" CONTENT="micro application, librairies
informatique, livres informatique, logiciel architecture,
architecture 3D, logiciel decoration, logiciel gravure, papier
imprimante, papier photo, impression papier, telechargement
logiciels, jeux cartes, formation web, cartes visites, achat
livres">
<META NAME="Author" CONTENT="Micro Application">
<LINK ... >
<SCRIPT>...</SCRIPT>
</HEAD>
```

Élément BODY

L'élément BODY renferme l'essentiel : le corps du document. Tout son contenu est visible dans le navigateur : texte, graphismes, liens hypertextes, etc. La seule façon de masquer une information consiste à la placer dans une balise de commentaire.

1. Revenez au document précédemment créé (*pageacc1_0.html*) s'il est toujours ouvert. Sinon, ouvrez le Bloc-Not0es, puis ouvrez le fichier.
2. Placez-vous après la balise </HEAD>, appuyez sur ↵, puis débutez le corps du document en saisissant <BODY>. Comme

précédemment, refermez l'élément `BODY` en appuyant sur ↵ puis en saisissant `</BODY>`.

Comme cela a été souligné, l'élément `BODY` de votre page Web ne doit pas être dépourvu de structure et de forme : tout « bon » document doit posséder une structure organisée. Traditionnellement, la structure se compose de titres de niveaux divers et de paragraphes de texte.

Titres : élément Hn

Les titres sont obtenus en HTML à l'aide de l'élément `Hn`, où `n` est un chiffre compris entre 1 et 6, correspondant au niveau du titre : `H1` est un titre principal (souvent identique au contenu de l'élément `TITLE`), `H2` un sous-titre, etc.

3 Pour commencer une page « bien structurée », débutez par un titre de niveau 1. Placez-vous à la fin de la balise `<BODY>`, puis appuyez sur ↵. Saisissez `<H1></H1>` (pour refermer immédiatement l'élément).

4 Placez-vous entre les deux balises, et saisissez ensuite le titre de la page : `Ma page d'accueil` (sans guillemets). Votre site Web possède maintenant un titre de premier niveau.

5 Créez un titre de niveau 2 : placez-vous à la fin de la ligne précédente, appuyez sur ↵ puis saisissez `<H2></H2>` (pour refermer immédiatement l'élément). Placez-vous entre les deux balises, et saisissez ensuite : `Bienvenue sur mon site`. Appuyez sur ↵.

6 Toujours par sécurité, enregistrez à nouveau votre travail sous le même nom.

Un titre peut se trouver n'importe où dans l'élément `BODY`, c'est-à-dire dans le corps du document. Il s'agit d'un élément bloc, ce qui signifie qu'un saut de ligne est effectué avant et après l'élément.

Le tableau suivant détaille les différents éléments titre.

Tableau 3.1 : *Élément titre (Hn)*

Titre	Balise	Utilisation
Titre 1	`<H1>`	Titre du niveau supérieur : en général celui de la page ou un titre de chapitre.

Tableau 3.1 : *Élément titre (Hn)*		
Titre	**Balise**	**Utilisation**
Titre 2	`<H2>`	Titre de second niveau : section ou partie d’un document.
Titre 3	`<H3>`	Sous-sections d’un document : ces trois niveaux suffisent le plus souvent, en terme de structure.
Titre 4	`<H4>`	Peu utilisé pour la structure.
Titre 5	`<H5>`	Peu utilisé pour la structure, mais servant de façon abusive pour des avertissements en petits caractères, comme des informations légales ou de droits d’auteur.
Titre 6	`<H6>`	Sert souvent de façon abusive pour des avertissements en petits caractères, comme des informations légales ou de droits d’auteur.

Les titres constituent un élément essentiel de la structure d’un document. De nombreux auteurs les emploient toutefois comme raccourcis de mise en forme : au lieu de modifier directement la police d’un texte (ce qui est d’ailleurs à déconseiller), ils l’incorporent dans un élément titre offrant un aspect similaire à celui qu’ils veulent obtenir.

Techniquement parlant, cette utilisation est incorrecte, tous les navigateurs n’affichant pas les titres de la même façon. Il est toutefois exact que la plupart des navigateurs affichent les titres des trois niveaux supérieurs (1 à 3) en polices de grandes tailles, tandis que les autres niveaux (5 et 6) sont affichés en polices plus petites, souvent en gras.

Pour obtenir une structure parfaite, il est donc déconseillé de recourir aux titres uniquement pour obtenir l’aspect souhaité : les niveaux de titre ne doivent pas présenter d’interruption.

```
<H1>…</H1>
   <H2>…</H2>
      <H3>…</H3>
      …
      <H3>…</H3>
      …
   <H2>…</H2>
…
```

est correct.

```
<H1>…</H1>
   <H2>…</H2>
      <H4>…</H4>
```

```
        …
        <H4>…</H4>
        …
    <H2>…</H2>
…
```

ne l’est pas.

Si vous voulez obtenir l’aspect d’un titre de niveau 4 pour votre titre de niveau 3, modifiez le style de l’élément H3, de préférence à l’aide d’une feuille de style. N’employer les titres qu’à des fins structurelles. Si votre public n’utilise que des navigateurs textuels (comme Lynx), ou si votre site s’adresse à des personnes souffrant de handicaps, les titres doivent être utilisés à bon escient. Cela épargnera à vos visiteurs employant des navigateurs en mode texte (comme Lynx) de sérieuses difficultés lorsque leur navigateur « non standard » affichera (ou de lire à haute voix) votre texte de façon incohérente. Les convertisseurs texte-parole emploient en effet souvent une inflexion particulière ou un signal lorsqu’ils rencontrent un élément HTML particulier, comme justement un titre.



Titres et moteurs de recherche

Les principaux moteurs de recherche se servent désormais principalement de la structure du document définie par les titres pour cataloguer un document. Initialement, ils n’avaient recours qu’aux éléments TITLE ou META, mais ils se sont rapidement rendu compte que des concepteurs peu scrupuleux surchargeaient ces éléments de mots-clés n’ayant rien à voir avec le contenu réel du site, simplement pour attirer plus de visiteurs.

En parcourant les titres, un moteur de recherche peut appréhender la structure du document et ainsi le cataloguer avec une meilleure précision. Si les éléments essentiels de votre site apparaissent dans vos titres, votre page Web possède des chances maximales de se retrouver classée dans la catégorie adéquate, là où la cherche le public visé.



Titres de documents et outils de vérification

Les outils de vérification HTML (comme le validateur W3C disponible à l’adresse http://validator.w3.org/) se servent également des titres pour reconstituer la structure du document. Cela sert généralement à l’auteur de la page : il peut regarder si la structure du document est bien représentée par les titres, ou si la page gagne à être réorganisée.

Paragraphes de texte : élément P

Toute page Web est essentiellement composée de texte : ne laissez jamais les graphismes attrayants, les animations époustouflantes et autres compléments vous masquer ce fait. Les gens vont sur le Web pour trouver des informations, pas pour admirer l’image d’un avion de ligne traversant l’écran.

L’élément texte fondamental est le paragraphe défini par l’élément `P`. Il n’exige pas de fermeture (la balise `</P>` est facultative), bien que cela soit préférable pour créer une page structurée possédant des exigences de validation stricte.

Un simple retour chariot ne possède aucune signification en HTML : le navigateur n’en tient pas compte et ne le remplace même pas par un simple espace. Pour commencer un nouveau paragraphe, veillez à bien employer la balise `<P>`.

1 Revenez au document *pageacc1_0.html*.
2 Créez un paragraphe : placez-vous sur la ligne vierge en dessous du titre de niveau 2, puis saisissez `<P></P>` (pour refermer immédiatement l’élément). Placez-vous entre les deux balises, et saisissez ensuite une phrase quelconque, par exemple : `Vous trouverez sur ce site des informations sur ma région, ma famille et mes passions.`
3 Nous allons nous arrêter là dans un premier temps. Enregistrez à nouveau votre fichier, toujours sous le nom *pageacc1_0.html*.

Ce que vous avez saisi jusque-là devrait ressembler au listing suivant.

Listing 3-3 : Première page d’accueil.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>"Ma page d’accueil"</TITLE>
<META name="author" content="mon nom">
<META name="version" content="1.0">
</HEAD>
<BODY>
<H1>Ma page d’accueil</H1>
<H2>Bienvenue sur mon site.</H2>
<P>Vous trouverez sur ce site des informations sur ma région,
ma famille et mes passions.</P>
</BODY>
</HTML>
```

Il est temps d'examiner cette page à l'aide de votre navigateur. Ouvrez celui-ci, puis naviguez jusqu'au fichier *pageacc1_0.html*. Vous devriez voir quelque chose de similaire à ce qui est présenté dans les figures qui suivent : aucune différence n'apparaît que vous utilisiez Internet Explorer 6 ou Firefox. Remarquez que le contenu de l'élément `TITLE` apparaît dans la barre de titre de la fenêtre du navigateur, le reste du contenu de l'élément `HEAD` restant invisible : seul le contenu de l'élément `BODY` est affiché.



Figure 3.8 : *L'aspect de cette page avec Internet Explorer 6*



Figure 3.9 : *L'aspect de cette page avec Firefox*

Difficile de faire plus simple ! Difficile aussi de faire plus laid, hélas… En outre, pour le moment, il n'existe aucune relation entre cette page d'accueil et les autres pages du site, que nous avions pourtant prévues dans notre conception initiale. Nous allons d'abord ajouter les liens vers les autres pages, puis améliorer un peu l'aspect de la page. Nous créerons ensuite les pages secondaires.

Lien hypertexte : élément A

Pour effectuer un lien entre une partie quelconque (ou presque) de votre page et une autre ressource Internet pouvant se trouver n'importe où (sur la même page, sur une page différente située sur le même site, sur une page située sur un autre site, vers un fichier ou une image situé à un emplacement quelconque), il est fait appel à l'élément `A` (*anchor*, signifiant « ancre »), un élément de type ligne (sans saut de ligne après l'élément).

Les liens hypertextes sont ce qui fait le World Wide Web (ou WWW, ou toile d'araignée mondiale). Ce sont eux qui permettent de passer de page en page et de site en site, d'un bout à l'autre de la planète. En vérité, cet élément, pourtant simple d'utilisation, est probablement le plus important de tout le langage HTML, étant donné sa puissance intrinsèque.

La syntaxe de l'élément `A` est la suivante :

```
<A [name="nom_lien"] [id="id_lien"] href="emplacement de la
ressource">contenu servant de lien</A>
```

Chaque élément `A` définit une ancre. Au sein de cet élément, l'attribut `href` fait de cette ancre la source d'exactement un lien, tandis que le contenu de l'élément `A` définit la position de l'ancre. L'attribut `name`, facultatif, nomme l'ancre (pour qu'elle puisse être la destination de zéro ou plusieurs liens. Il en est de même de l'attribut `id`).



Valeur d'attribut

Les auteurs peuvent également créer un élément `A` qui ne spécifie aucune ancre (qui ne spécifie aucun attribut `href`, `name` ou `id`). Les valeurs de ces attributs peuvent être fixées par la suite au moyen de scripts.

Attribut href et URI

La valeur de l'attribut `href`, *emplacement_de_la ressource*, porte le nom d'*identifiant de ressource uniforme* ou URI (*Universal Resource Identifier*), terme désormais préféré à l'ancien URL ou *localisateur de ressource uniforme (*URL, *Universal Resource Locator*). Le rôle d'un URI est capital : chaque ressource disponible sur le Web (document HTML,

image, séquence vidéo, programme, etc.) possède une adresse représentée sous la forme d'un URI.

Un URI est formé de trois parties distinctes :

1. Le nom du protocole servant à accéder à la ressource.
2. Le nom de la machine hébergeant la ressource.
3. Le nom de la ressource en question, indiqué sous la forme d'un chemin d'accès.

Considérez comme exemple l'URI qui désigne la page des rapports techniques (TR, *Technical Reports*) du W3C :

```
http://www.w3.org/TR
```

Cet URI peut être compris comme suit : il est disponible *via* le protocole HTTP, hébergé sur la machine nommée « www.w3.org », et accessible par le chemin « /TR ».



HTTP

HTTP est le sigle de *HyperText Transfer Communication Protocol*. C'est le protocole, ou méthode de communication, employé par un navigateur Web pour requérir une ressource Web auprès d'un serveur. Le terme `http://` précède toujours une adresse URI ; il est placé avant les noms du domaine et de la machine. Internet dispose d'autres protocoles de communication : HTTP sécurisé (`https`), courrier électronique (`mailto`) et protocole de transfert de fichiers (`ftp`) sont tous d'autres exemples de protocoles largement utilisés, que vous rencontrerez sans aucun doute.

Il s'agit ici d'une adresse URI dite *exclusive*, utilisée pour les sites situés à l'extérieur de votre propre domaine.

Il existe un autre type d'adresse URI, l'URI *relative*, qui localise un document selon sa relation avec le document actuellement visualisé. Si le document courant est dans un dossier, et que le document lié (nommé pour l'exemple *cible.html*) est dans ce même dossier, il peut être relié en utilisant simplement :

```
<A href="cible.html">
```

Remarquez, pour les URI locaux, l'absence du préfixe `http://`.

Au sein de la chaîne de texte du chemin, vous pouvez employer les classiques modificateurs :

`../` demande de monter d'un niveau dans l'arborescence par rapport au document actif :

```
<A href="../cible.html">
```

Pour atteindre un sous-dossier, saisissez son nom puis une barre oblique et enfin le nom du fichier, comme dans :

```
<A href="dossier/cible.html">
```

Vous pouvez les combiner à votre guise :

```
<A href="../dossier/cible.html">
```

Une barre oblique placée au début de l'URI signifie qu'il faut remonter à la racine du domaine :

```
<A href="/cible.html">
```

Les agents utilisateurs restituent généralement les liens de façon à les mettre en évidence pour les utilisateurs (soulignage, vidéo inverse, etc.). La restitution exacte dépend de l'agent utilisateur. La restitution peut varier si l'utilisateur a ou non déjà visité le lien. De façon générale, le pointeur se transforme en main lorsqu'il se trouve sur un lien.



Figure 3.10 : La plupart de navigateurs signalent un lien, et l'affichent de façon différente lorsqu'il a déjà été visité (ici, Logiciels en téléchargement).

Attributs name et id

Ces attributs spécifient un nom ou un identifiant pour l'ancre, de façon à ce que celle-ci puisse être ciblée par un autre lien. Ils partagent le même espace de noms, ce qui signifie qu'ils ne peuvent pas tous deux définir

une ancre avec le même nom dans le même document. Quand les deux attributs sont utilisés sur un même élément, leurs valeurs doivent être identiques.

Voici une ancre nommée :

```
<A name="intro">Introduction</A>
```

Pour y faire référence depuis une autre ancre, vous procédez comme suit :

```
<A href="#intro">Introduction</A>
```

Ce qui pourrait être mis en œuvre comme suit :

```
<H1>Sommaire</H1>
<P><A href="#intro">Introduction</A><BR>
<A href="#historique">Historique</A><BR>
<A href="#chap1">Chapitre 1 </A><BR>
...le reste du tableau des matières...
...le corps du document...
<H2><A name="intro">Introduction</A></H2>
...contenu...
<H2><A name="historique">Historique</A></H2>
...contenu...
<H3><A name="histo1">Au commencement</A></H3>
...contenu...
```

Nous présentons ici l’attribut `name` employé avec un élément `a`, mais une ancre sert en réalité rarement de cible à une autre ancre. L’attribut `name` n’est employé que pour relativement peu d’éléments. L’attribut `id` joue en revanche un rôle analogue et peut être employé avec tous les éléments sauf `BASE`, `HEAD`, `HTML`, `META`, `SCRIPT`, `STYLE` et `TITLE`. L’exemple précédent serait ainsi plus probablement écrit :

```
<H1>Sommaire</H1>
<P><A href="#intro">Introduction</A><BR>
<A href="#historique">Historique</A><BR>
<A href="#chap1">Chapitre 1 </A><BR>
...le reste de la table des matières...
...le corps du document...
<H2 id="intro">Introduction</H2>
...contenu...
<H2 id="historique">Historique</H2>
...contenu...
<H3 id="histo1">Au commencement</H3>
...contenu...
```

Notez qu’une ancre (ou lien) ainsi employée porte fréquemment le nom de *signet*, pour la différencier d’un lien menant vers une autre page Web.

Il est également possible de concevoir un lien hypertexte qui déclenche l'envoi d'un courrier électronique lorsqu'un utilisateur clique dessus. Au lieu de `http://`, utilisez `mailto:` suivi de l'adresse électronique de destination :

```
<A "mailto:mon_nom@monFAI.fr">
```

Remarquez, pour un lien de message électronique, l'absence de barre oblique.

Ajoutez maintenant des ancres à votre page d'accueil.

1 Ouvrez le Bloc-Notes, puis le fichier précédemment créé (*pageacc1_0.html*).

2 Vous allez modifier la version en entrant comme suit la balise `META` stipulant la version :

```
<META name="version" content="1.3.2">
```



Convention sur les numéros de version

Traditionnellement, un numéro de version comporte deux ou trois nombres séparés par des points. Le premier nombre, à gauche, correspond au numéro de version majeur : il reste identique tant que des modifications radicales n'ont pas été apportées. Le second numéro correspond au numéro de version mineur : des modifications plus secondaires. Le troisième nombre, s'il est présent, correspond généralement à des modifications très secondaires, souvent liées à l'éradication de bogues ou la correction d'erreurs.

Dans le cadre de ce livre, nous employons la convention suivante, très proche de la convention académique :

- le premier nombre correspond à la version majeure ;
- le second correspond au chapitre concerné ;
- le troisième, à l'ordre d'apparition du code dans le chapitre.

La nouvelle version de notre page d'accueil est donc la 1.3.2.

Vous allez maintenant créer des ancres. Celles-ci vont pour le moment pointer vers des pages inexistantes, mais cela n'a pas encore d'importance.

3 Insérez l'élément lien hypertexte avant le mot ou la phrase et n'oubliez pas de fermer l'élément `A` après le mot servant de lien :

```
<P>Vous trouverez sur ce site des informations sur ma
<A href="region.html">région</A>, ma <A href="famille.html">
famille</A> ; et mes <A href="passions.html">passions</A>.</P>
```

- Remarquez qu'il est employé ici un lien relatif : les fichiers *region.html*, *famille.html* et *passions.html* seront placés dans le même dossier que la page d'accueil.

4 Ajoutez un nouveau lien permettant de vous envoyer un message. Placez-vous après la balise </P>, appuyez sur ↵ et saisissez sur la nouvelle ligne :

```
<P align="center"><A href="mailto:votre_nom@votre_FAI">
Ecrivez-moi !</A>.</P>
```

- *votre_nom@votre_FAI* correspond bien sûr à votre adresse électronique.

5 Enregistrez votre fichier sous le nom *pageacc1_3_2.html*, mais ne fermez pas le Bloc-Notes. Le code complet de cette page est présenté dans le listing qui suit.

Listing 3-4 : Page d'accueil modifiée (version 1.3.2)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>"Ma page d'accueil"</TITLE>
<META name="author" content="mon nom">
<META name="version" content="1.3.2">
</HEAD>
<BODY>
<H1>Ma page d'accueil</H1>
<H2>Bienvenue sur mon site.</H2>
<P>Vous trouverez sur ce site des informations sur ma
<A href="region.html">région</A>, ma <A href="famille.html">
famille</A> et mes <A href="passions.html">passions</A>.</P>
<P><A href="mailto:votre_nom@votre_FAI">Ecrivez-moi !</A>.</P>
</BODY>
</HTML>
```

Examinez cette nouvelle page dans votre navigateur. Ouvrez celui-ci, puis naviguez jusqu'au fichier *pageacc1_3_2.html*. Vous devriez voir quelque chose de similaire à ce qui est présenté dans la figure suivante.

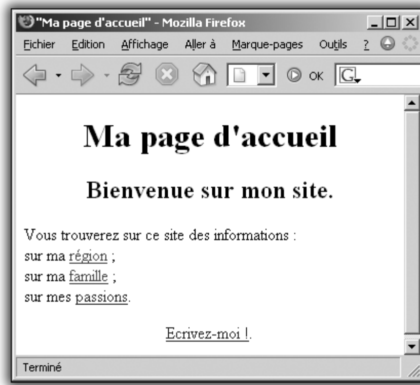


Figure 3.11 :
Aspect de la page Web modifiée

Essayez de cliquer sur un des liens : vous obtenez un message d'erreur, variable selon le navigateur. C'est normal : les pages secondaires n'existent pas encore ! Vous y remédierez sous peu.

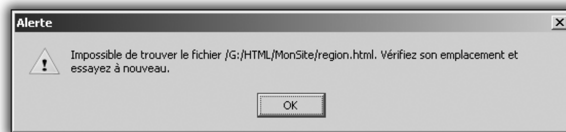


Figure 3.12 :
Message d'erreur indiquant un lien rompu

Élément saut de ligne BR

Nous avons dit que les éléments `Hn` et `P`, déjà examinés, étaient des éléments de type bloc, qui débutent sur une nouvelle ligne et insèrent après eux un saut de ligne. Nous avons également souligné que HTML ne reconnaissait pas les sauts de ligne insérés dans le contenu d'un élément. Il est pourtant parfois nécessaire d'effectuer manuellement un passage à la ligne sans pour autant changer d'élément (de bloc), par exemple pour posséder un titre réparti sur deux lignes ou pour diviser un paragraphe trop long. Cela s'effectue en insérant un élément vide `BR`, de type ligne :

1. Revenez dans le Bloc-Notes au fichier précédent (*pageacc1_3_2.html*).
2. Vous allez encore modifier la version. Entrez comme suit la balise META stipulant la version :

```
<META name="version" content="1.3.3">
```

3 Vous allez maintenant insérer des sauts de ligne entre les différents thèmes dans le (seul) paragraphe de texte. Commencez par :

```
<P>Vous trouverez sur ce site des informations :<BR>
sur ma <A href="region.html">région</A> ;<BR>
sur ma <A href="famille.html">famille</A> ;<BR>
sur mes <A href="passions.html">passions</A>.</P>
```

- Remarquez que le texte a été un peu modifié. Les passages à la ligne employés dans le code présenté (après les balises `<BR>`) permettent uniquement d'améliorer sa lisibilité : ils sont ignorés par HTML. Tout le contenu de l'élément `P` aurait pu être saisi sur la même ligne.

Vous pouvez insérer un saut de ligne où vous voulez dans l'élément `BODY` pour obtenir un saut de ligne à l'écran : entre deux éléments bloc, entre deux éléments ligne, au sein d'un contenu texte, etc.



XHTML

Si vous souhaitez être compatible XHTML, l'élément vide `BR` doit être codé `<br />`.

Attributs d'alignements

Pour encore améliorer l'aspect d'un titre (ou d'un paragraphe), il serait utile de pouvoir agir sur l'alignement.

HTML dispose d'un attribut d'alignement applicable à de nombreux éléments, `align`.

Vous allez un peu améliorer l'aspect de la première page en jouant sur l'alignement.

1 Revenez dans le Bloc-Notes au fichier précédent (*pageacc1_3_3.html*).

2 Centrez les deux titres, à l'aide de l'attribut `align` :

```
<H1 align="center">Ma page d'accueil</H1>
<H2 align="center">Bienvenue sur mon site.</H2>
```

3 Modifiez le lien qui permet de vous envoyer un message. Ajoutez ce qui suit :

```
<P align="center"><A href="mailto:votre_nom@votre_FAI">
Ecrivez-moi !</A>.</P>
```

4 Enregistrez votre fichier sous le nom *pageacc1_3_3.html*. Le code complet de cette page est présenté dans le Listing 3.4.

5 Pour la bonne forme, nous allons également créer le squelette des autres pages. En partant du fichier actuellement ouvert, commencez par modifier l'élément `TITLE` :

```
<TITLE>"Ma région "</TITLE>
```

6 Modifiez ensuite le titre et le sous-titre …

```
<H1 align="center">Ma région </H1>
<H2 align="center">...à complèter.</H2>
```

7 … puis le paragraphe de texte :

```
<P>... à compléter.</P>
```

8 Vous pouvez à votre guise conserver ou supprimer la ligne établissant un lien vers un message électronique. En revanche, il est capital de toujours prévoir sur une page secondaire un lien vers la page d'accueil principale. Ajoutez donc la ligne suivante, juste avant la balise `</BODY>` :

```
<P align="center"><A href="pageacc1_3_3.html">Retour vers
la page d'accueil</A></P>
```

9 Enregistrez ce fichier sous le nom *region.html*.

10 Créez ensuite les deux autres pages (*famille.html* et *passions.html*), en modifiant comme il se doit le contenu des éléments `TITLE` et `H1`.

L'attribut `align` possède plusieurs valeurs possibles, dont la signification est présentée dans le Tableau 3.2. Remarquez toutefois que son emploi avec des titres et paragraphes est désormais déconseillé : mieux vaut recourir aux feuilles de style, que vous découvrirez dans le Chapitre 8.

Tableau 3.2 : *Valeurs d'alignement des paragraphes*

Attribut de P	Alignement de texte
<P align="left">	Alignement à gauche.

Tableau 3.2 : *Valeurs d'alignement des paragraphes*	
Attribut de P	**Alignement de texte**
<P align="right">	Alignement sur la marge de droite, sans justification à gauche : celle-ci est plutôt désordonnée, mais cette disposition est pratique dans le cas d'images et de colonnes de texte.
<P align="center">	Le texte est centré.
<P align="justify">	Le texte est justifié, comme avec un traitement de texte.

Listing 3-5 : Page d'accueil modifiée (version 1.3.3).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>"Ma page d'accueil"</TITLE>
<META name="author" content="mon nom">
<META name="version" content="1.3.3">
</HEAD>
<BODY>
<H1 align="center">Ma page d'accueil</H1>
<H2 align="center">Bienvenue sur mon site.</H2>
<P>Vous trouverez sur ce site des informations :<BR>
sur ma <A href="region.html">région</A> ;<BR>
sur ma <A href="famille.html">famille</A> ;<BR>
sur mes <A href="passions.html">passions</A>.</P>
<P align="center"><A href="mailto:votre_nom@votre_FAI">
Ecrivez-moi !</A>.</P>
</BODY>
</HTML>
```

Examinez votre nouvelle page d'accueil dans votre navigateur. Ouvrez celui-ci, puis naviguez jusqu'au fichier *pageacc1_3_3.html*.



Figure 3.13 : *Page d'accueil*

Cliquez sur un lien : vous accédez à l'une des nouvelles pages.



Figure 3.14 :
Une des pages secondaires

Revenez à la page d'accueil, testez les autres liens, puis, revenu sur la page d'accueil, testez le lien d'envoi de courriel : votre logiciel de messagerie s'ouvre.

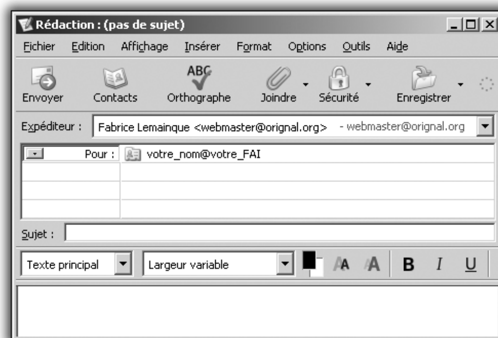


Figure 3.15 :
Logiciel de messagerie lancé à partir de la page

3.3. Résumé

- L'instruction `!DOCTYPE` permet de stipuler la DTD (*Document Type Definition*) employée pour le document HTML.
- Un document HTML se compose d'un élément parent `HTML` qui contient des éléments enfants `HEAD` et `BODY`.
- La seule information affichée par le navigateur appartenant à l'élément `HEAD` est le contenu de l'élément `TITLE`, figurant dans la barre de titre de la fenêtre. Les éléments `META` de l'élément

`HEAD` servent à fournir des méta-informations employées par certains agents utilisateurs (comme les robots des moteurs de recherche), dont les mots-clés, une description, l'auteur, la version et le générateur de la page.

- L'élément `BODY` constitue le corps du document : tout son contenu est visible dans le navigateur.
- Ce contenu doit être structuré de façon logique. HTML propose les éléments `Hn` pour des titres de niveaux différents, ainsi que l'élément `P` pour un paragraphe de texte.
- Les liens hypertextes sont créés à l'aide de l'élément `A` suivi de l'attribut `href` stipulant la cible du lien. Cet attribut `href` est suivi d'un nom de protocole, comme `http:/` ou `mailto:`, spécifiant le type de cible et l'action à suivre.
- Un lien hypertexte peut être absolu ou relatif.
- Un saut de ligne est obtenu à l'aide de l'élément vide `BR`.
- Les titres et les paragraphes admettent l'emploi de l'attribut `align` pour obtenir un alignement particulier. Cet attribut est toutefois désormais déconseillé : mieux vaut recourir aux feuilles de style.

Chapitre 4

Listes et tableaux

Listes 73
Tableaux 89
Résumé 133

Une page Web est composée avant tout de texte. Toutefois, toutes les données et informations placées sur un site Web doivent passer par un filtre final avant de pouvoir être lues par un être humain : elles doivent être affichées à l'écran.

La lecture à l'écran, malgré les progrès réalisés par les concepteurs d'écrans, reste plus malaisée que la lecture d'un document imprimé ou projeté. La majorité des internautes rechignent à lire de longs textes en ligne, préférant souvent les imprimer pour les relire à tête reposée.

Une conception intelligente d'un site Web doit prendre en compte ces contraintes inévitables. Malgré son importance, mieux vaut limiter le texte. Aérez-le toujours au maximum. Recourrez aux espaces vierges pour reposer la vue.

Voici quelques règles simples :

- Placez toujours les informations importantes en haut de l'écran.
- Pour être efficace, un écran (une page) ne devrait jamais comporter plus de sept informations distinctes.
- Seuls les 100 premiers mots permettent à l'utilisateur de décider si le document est intéressant. Un document destiné à un public général ne devrait jamais excéder 500 mots.

Cela ne veut pas dire que les longs articles sont proscrits sur le Web, mais mieux vaut les rendre accessibles à l'aide d'un lien explicite, par exemple, « *Exemple extrait du Chapitre 1* » ou « *Table des matières détaillée* ».

Si ce chapitre va vous aider à afficher votre texte de façon à ce qu'il soit facilement lisible et compréhensible, il reste de votre ressort de faire en sorte que le sujet soit intéressant et susceptible d'attirer le public concerné !

Les pages réalisées jusqu'ici sont d'une extrême simplicité. HTML propose de nombreux autres éléments structuraux pour améliorer tant la structure que, par la suite, le rendu ou l'affichage des pages.

Au cours de ce chapitre, nous allons examiner les listes et les tableaux des éléments fondamentaux de toute page Web réussie.

Éléments et attributs étudiés dans ce chapitre :

```
OL (type, start, type), LI (type, value)
UL (type), LI (type)
DL, DT, DD
TABLE (summary, align, border, frame, rule, cellspacing,
       cellpadding, width)
CAPTION
TR, TD, TH (align, valign, width, rowspan, colspan)
THEAD, TBODY, TFOOT
COLGROUP, COL (span, width)
```

4.1. Listes

Il existe en HTML trois types de listes qui permettent d'apporter un peu plus de structure à un document :

- Les listes ordonnées ou numérotées.
- Les listes non ordonnées ou listes à puces.
- Les listes de définitions.

Il est également possible d'imbriquer des listes, même de types différents.

Liste ordonnée : éléments OL et LI

Une liste numérotée (ou ordonnée) incrémente automatiquement chaque élément de la liste. Ce système de numérotation automatique est très utile, car il simplifie d'autant la saisie.

Créer une liste numérotée est d'une extrême simplicité. Vous débutez la liste en ouvrant l'élément liste OL (*Ordered List*, liste ordonnée), puis poursuivez par chaque membre de la liste, précédé de la balise d'élément de liste, <LI>. La balise de fermeture </LI> est facultative mais recommandée. En revanche, l'élément liste lui-même doit impérativement être fermé.

```
<OL>
<LI>Premier élément</LI>
<LI>Deuxième élément</LI>
...
<LI>Dernier élément</LI>
</OL>
```

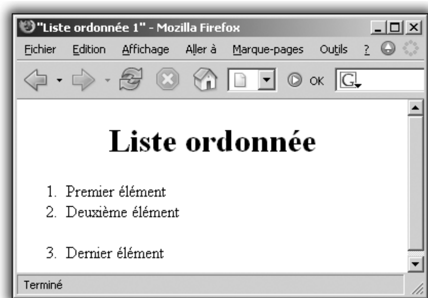


Figure 4.1 :
Exemple de liste ordonnée



Test

Les fragments de code ainsi présentés sont des extraits d'une page complète. Si vous souhaitez les tester, le mieux est de construire une page type nommée par exemple *test.html*, avec un squelette de page (instruction `!doctype`, éléments `HTML`, `HEAD` et `BODY`) , et de saisir simplement dans l'élément `BODY` l'extrait de code proposé.

L'élément `OL` peut posséder un argument `start`, spécifiant le numéro commençant la séquence. Par exemple, avec `<OL start="2">`, le premier membre de la liste porte le numéro 2.

Les éléments `LI` et `OL` peuvent posséder un attribut `type`, qui permet de définir le mode de numérotation de la liste. Employé pour un élément `LI` individuel, il ne s'applique qu'à cet élément. Inséré en revanche à l'intérieur de la balise `<OL>`, il agit sur la totalité de la liste. Voici les valeurs possibles pour l'attribut `type` :

- `1` : Chiffres arabes (valeur par défaut)
- `a` : Lettres minuscules
- `A` : Lettres majuscules
- `i` : Chiffres romains en minuscules
- `I` : Chiffres romains en majuscules

`value`, employé uniquement avec un élément `LI`, réinitialise l'ordre de la séquence. Ainsi, avec `<LI value="8">`, le membre suivant possédera automatiquement une `value` de 9.

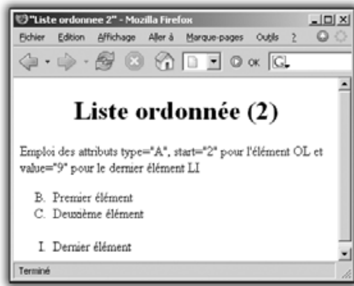


Figure 4.2 :
Exemple de liste ordonnée, avec emploi des attributs type="A", start="2" pour l'élément OL et value="9" pour le dernier élément LI.

L'emploi des attributs `start`, `type` et `value` est désormais déconseillé au profit des feuilles de style.



Combiner les attributs

Il est possible de combiner les attributs `start` et `type` : le premier membre de la liste apparaît avec la valeur `start` dans le `type` spécifié. Ainsi, `<OL type="I" start="4">` aboutit-il à un premier membre désigné par "`IV`".

Liste à puces : éléments UL et LI

Une liste à puces ressemble énormément à la précédente, si ce n'est que la numérotation est remplacée par des puces. Une liste à puces est un élément `UL` (*Unordered List*, liste non ordonnée) qui renferme un nombre quelconque d'éléments `LI`. Comme pour une liste ordonnée, l'élément `UL` doit impérativement être fermé.

```
<UL>
<LI>Premier élément</LI>
<LI>Deuxième élément</LI>
...
<LI>Dernier élément</LI>
</UL>
```



Figure 4.3 :
Exemple de liste à puces

Vous pouvez modifier le style des puces grâce à l'attribut `type`. Là aussi, il peut s'appliquer à un élément `LI` individuel ou, inséré dans la balise `<UL>`, agir sur la totalité de la liste. Voici les valeurs possibles pour l'attribut `type`, lorsqu'il est employé dans une liste non ordonnée :

Tableau 4.1 : *Valeurs possibles de l'attribut type*

Valeur	Signification
`Disc`	Rond plein (valeur par défaut)
`Square`	Carré plein
`Circle`	Cercle vide

L'emploi de l'attribut `type` est désormais déconseillé au profit des feuilles de style.

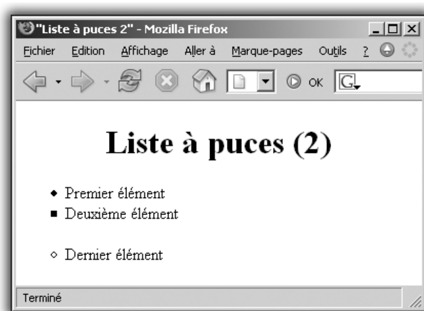


Figure 4.4 : *Exemple de liste à puces, avec emploi des trois valeurs possibles pour l'attribut type.*

Les listes sont très agréables en HTML : elles sont d'emploi facile tout en étant d'une extrême souplesse. Il est très facile de transformer une liste numérotée en liste à puces et inversement, simplement en modifiant les balises de début et de fin. Attention toutefois aux surprises si vous avez employé les attributs déconseillés, les valeurs licites pour ces attributs étant différentes selon le type de liste !

Liste de définitions : éléments DL, DT et DD

Une liste de définitions est l'équivalent d'un tableau de deux colonnes, la première renfermant les mots ou expressions à définir et la seconde les définitions elles-mêmes.

Il est possible de créer une liste de définitions à l'aide de l'élément DL (*Definition List*, liste de définitions), dont le contenu est composé d'éléments DT (*Definition Term*, terme de définition) et d'éléments DD (*Definition Description*, description de définition) correspondants. Voici un exemple de code, avec la copie d'écran correspondante.

```
<H1 align="center">Eléments d'une liste de définitions</H1>
<DL>
<DT>DL</DT>
<DD>Elément liste de définition.</DD>
<DT>DT</DT>
<DD>Elément terme de définition</DD>
<DT>DD</DT>
<DD>Elément description de la définition</DD>
</DL>
```

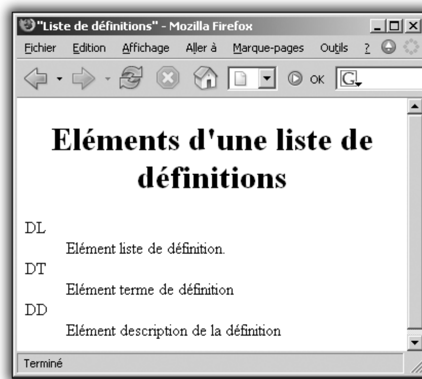


Figure 4.5 :
Liste de définitions

Vous avez découvert les listes. Ne trouvez-vous pas que notre page d'accueil présente ce qui ressemble furieusement à une liste ? Autant la modifier…

1 Revenez dans le Bloc-Notes au fichier précédent (*pageacc1_3_3.html*).

2 Comme d'habitude, vous modifiez la version. Entrez comme suit la balise META stipulant la version :

```
<META name="version" content="1.4.1">
```

3 Placez-vous à la fin de la ligne

```
<P>Vous trouverez sur ce site des informations :<BR>
```

et remplacez la balise
 par une balise </P>, puis appuyez sur ↵ pour créer une nouvelle ligne.

4 Saisissez sur celle-ci `<UL>` pour débuter une liste à puces.

5 Placez-vous au début de la ligne

```
sur ma <A href="region.html">région</A> ;<BR>
```

Saisissez `<LI>`, puis remplacez la balise `<BR>` par `</LI>`.

6 Faites de même pour la ligne suivante. Saisissez au début de la troisième ligne `<LI>`, puis remplacez la balise de fermeture `</P>` par `</LI>`, appuyez sur ↵ puis saisissez `</UL>` pour fermer la liste. Ce fragment de code devrait désormais ressembler à ceci :

```
<P>Vous trouverez sur ce site des informations :</P>
<UL>
<LI>sur ma <A href="region.html">région</A> ;</LI>
<LI>sur ma <A href="famille.html">famille</A> ;</LI>
<LI>sur mes <A href="passions.html">passions</A>.</LI>
</UL>
```

7 Enregistrez votre fichier sous le nom *pageacc1_4_1.html*.

Examinez cette page à l'aide de votre navigateur. Son aspect est désormais plus agréable.

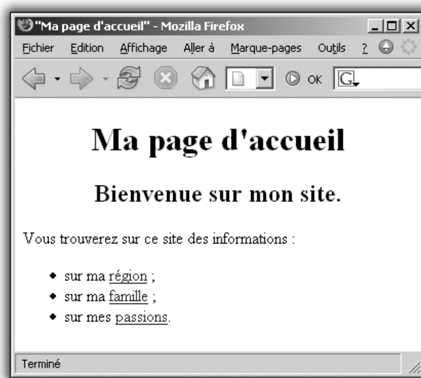


Figure 4.6 :
Page d'accueil avec liste à puces

Testez les liens. Tout va bien, mais lorsque vous voulez revenir à votre page d'accueil, vous obtenez une des anciennes versions. Pourquoi ? Tout simplement parce que vous n'avez pas modifié le nom de la cible (attribut `href`) dans les pages secondaires. Le lien stipule toujours :

```
<A href="pageacc1_3_3.html">Retour vers la page d'accueil</A>
```

Ce type d'erreur est particulièrement fréquent (autant qu'exaspérant !). Vous verrez par la suite une des façons de l'éviter, grâce notamment à

l'aide d'une barre de navigation, mais voici déjà quelques règles et astuces simples permettant d'éviter au moins partiellement ce piège :

1. Ne modifiez un nom de fichier comportant un numéro de version qu'en cas de modification au moins mineure (par exemple ici, le passage de la version 1.3 à la 1.4). Il est en principe inutile de noter ainsi les modifications réellement secondaires, qui ne sont souvent que des corrections de bogues. En revanche, pensez à actualiser le contenu de votre balise <META name ="version" content="numéro de version">.
2. Une bonne habitude à prendre consiste à archiver régulièrement tous les fichiers composant le site, et au minimum lors d'un changement de version mineure. Vous pouvez ainsi compacter, par exemple dans un fichier nommé ici *MonSitev1_3.zip*, tous les fichiers créés et employés jusque-là (sauf *pageacc1_4_1.html*).
3. Idéalement, cette archive (ou une copie) devrait être stockée à un autre emplacement : en cas d'incident, vous pourriez ainsi revenir à la version antérieure.
4. Supprimez ensuite de la racine du site les fichiers superflus (ici, les fichiers *pagacc1_3_x.html*).
5. Enfin, modifiez les fichiers présentant dans la nouvelle version des problèmes (liens ou autres) en corrigeant les erreurs.

Remarquez que c'est exactement la démarche que nous avons retenue, puisque les différentes archives *MonSite* correspondent à chaque chapitre.

Listes imbriquées

Il est très facile d'imbriquer des listes, même de types différents. Pour ce faire, débutez simplement une nouvelle liste plutôt que de créer un nouveau membre de liste. La nouvelle liste se comporte comme une subdivision du membre le précédant immédiatement. Voici un exemple :

Création d'une liste imbriquée avec un éditeur de texte

Listing 4-1 : Exemple de code de listes imbriquées

```
<H1>Arguments et valeurs possibles dans des listes</H1>
<OL type="I">
   <LI>Liste ordonnée</LI>
   <UL type="square">
      <LI>Elément OL</LI>
      <DL>
         <DT>start</DT>
         <DD>valeur de début de la numérotation</DD>
         <DT>type</DT>
         <DD>1,Chiffres arabes (valeur par défaut)<BR>
            a : Lettres minuscules<BR>
            A : Lettres majuscules<BR>
            I : Chiffres romains en minuscules<BR>
            I : Chiffres romains en majuscules</DD>
      </DL>
      <LI>Elément LI</LI>
      <DL>
         <DT>value</DT>
         <DD>valeur de réinitialisation de la numérotation</DD>
         <DT>type</DT>
         <DD>1,Chiffres arabes (valeur par défaut)<BR>
            a : Lettres minuscules<BR>
            A : Lettres majuscules<BR>
            I : Chiffres romains en minuscules<BR>
            I : Chiffres romains en majuscules</DD>
      </DL>
   </UL>
   <LI>Liste à puces</LI>
   <UL type="square">
      <LI>Elément UL</LI>
      <DL>
         <DT>type</DT>
         <DD> disc : Rond plein (valeur par défaut)<BR>
            square : Carré plein<BR>
            circle : Cercle vide</DD>
      </DL>
      <LI>Elément LI</LI>
      <DL>
         <DT>type</DT>
         <DD> disc : Rond plein (valeur par défaut)<BR>
            square : Carré plein<BR>
            circle : Cercle vide</DD>
      </DL>
   </UL>
</OL>
```

La figure suivante présente l'aspect de ce code vu dans un navigateur.

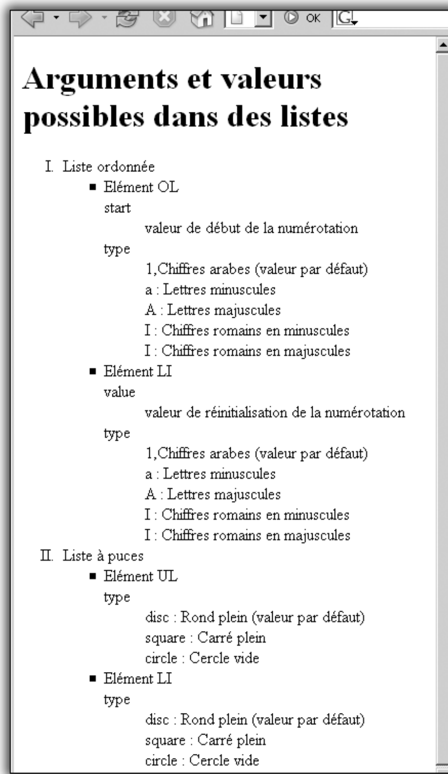


Figure 4.7 :
Listes imbriquées

Ce code présente plusieurs choses intéressantes. Remarquez d'abord la présence de numéros de ligne : ceux-ci ne sont présents que pour faciliter l'interprétation du code et ne doivent pas être saisis. En revanche, le code a été indenté pour mettre en évidence les imbrications : les éléments de même rang apparaissent les uns en dessous des autres, tandis que les passages à la ligne HTML (`<BR>`) ont été matérialisés à l'aide de retours chariot « manuels ». Comme cela a déjà été précisé, les espaces vides supplémentaires (espaces, tabulations, sauts de ligne) sont ignorés par HTML : ils ne servent qu'à simplifier la lecture du code. C'est une bonne habitude à prendre lors de la saisie manuelle de code. La plupart des éditeurs de code WYSIWYG accomplissent cela automatiquement.

Examinez le code plus en détail : après la ligne 1 (l'élément titre de premier niveau `H1`), il est débuté en ligne 2 une liste ordonnée à l'aide

de l'élément `OL`, dont l'attribut `type` stipule que la numérotation est faite en chiffres romains majuscules (`type="I"`).

Vient ensuite, ligne 3, un élément de ligne `LI`, puis, ligne 4, débute une liste à puces, l'élément `UL` possédant un attribut `type` fixé à `square`, pour définir une puce carrée pleine. Vient ensuite un élément de liste (`LI`), puis commence, ligne 6, une liste de définitions énumérant les attributs et les valeurs possibles pour cet élément.

Il suffit ensuite de fermer la liste de définitions (ligne 15), de placer un autre élément `LI` (pour l'élément `LI` de liste ordonnée, ligne 16) et de recommencer une liste de définitions. Nous fermons ensuite la liste de définitions (ligne 26), la liste à puces (ligne 27) et arrivons au second membre de la liste ordonnée (ligne 28). La suite reprend la structure présentée à partir de la ligne 4. La dernière liste de définitions est fermée ligne 43, la liste à puces ligne 44 et enfin la liste numérotée ligne 45. Ouf… Trois types différents de listes imbriquées, tout en jouant à l'aide de l'attribut `type` sur l'aspect de ces listes…

Cette structure demande la saisie d'une quantité de code importante. Il est donc intéressant de voir en quoi le recours à un éditeur HTML permet de faciliter les choses.

Création d'une liste imbriquée avec un éditeur HTML

Nous allons ici créer des listes imbriquées à l'aide de l'éditeur gratuit du W3C, Amaya.

1. Ouvrez Amaya (**Démarrer** > **Tous les programmes** > **Amaya** > **Amaya**).
2. Créez un nouveau document en choisissant **File** > **New** > **New XHTML 1.0 Transitional document**. Par défaut, il est proposé de créer le fichier dans le dossier d'installation d'Amaya : naviguez jusqu'à votre dossier *Mon site Web*, puis saisissez comme nom de document `ListesImbriquées2.html`.
3. Le document est de type XHTML 1.0. Transformez-le immédiatement en document HTML en choisissant **File** > **Change the Document Type** > **Change to HTML 4.01 Transitional**.

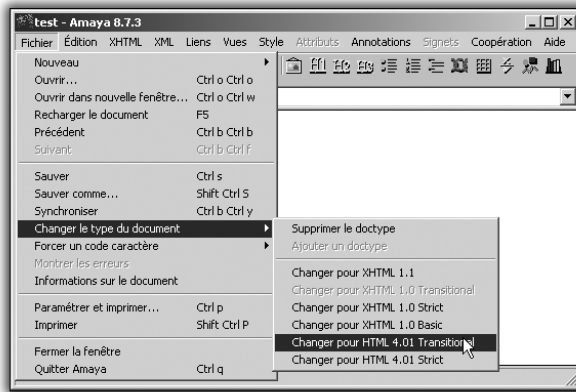


Figure 4.8 : *Modification du type de document avec Amaya*

4 Remarquez que la barre de titre affiche *no title*. Modifiez le titre du document. Choisissez **XHTML**> **Change title**, puis saisissez `Listes imbriquées`. La barre de titre affiche *Listes imbriquées*.

5 Créez un titre de niveau 1 : cliquez sur le bouton **H1**, puis saisissez `Arguments et valeurs possibles dans des listes`.

6 Débutez une liste numérotée en cliquant sur le bouton **Numbered list**.

7 Sélectionnez la ligne numérotée 1, puis choisissez **Edit** > **Insert**. Une seconde ligne numérotée 2 apparaît en dessous. Remarquez qu'avec Amaya, pour créer des listes imbriquées, mieux vaut définir la structure d'abord : nous créons les deux lignes de la première liste avant d'insérer les autres listes.

8 Placez-vous juste après 2, puis saisissez `Liste à puces`.

9 Placez-vous juste après 1, puis saisissez `Liste ordonnée`.

10 Débutez une liste à puces en cliquant sur le bouton **Bulleted list**.

11 Comme précédemment, cette liste à puces doit compter deux éléments : sélectionnez la première ligne, puis choisissez **Edit** > **Insert**.

12 Placez-vous sur la première ligne, puis saisissez `Elément OL`.

13 Débutez une liste de définitions en cliquant sur le bouton **Définition list**, puis saisissez `start`.

14 Choisissez **XHTML** > **List** > **Definition (dd)**, puis saisissez `valeur de début de la numérotation`.

15 Choisissez **XHTML** > **List** > **Term (dt)**, puis saisissez `type`.

16 Choisissez **XHTML** > **List** > **Definition (dd)**.

17 Saisissez `1 : Chiffres arabes (valeur par défaut)`, puis choisissez **XHTML** > **Break (br)**.

18 Répétez l'étape 13, en saisissant successivement `a : Lettres minuscules`, `A : Lettres majuscules` et `i : Chiffres romains en minuscules` en insérant après à chaque fois un saut de ligne.

19 Saisissez `I : Chiffres romains en majuscules`.



Chaînes de texte

Vous pouvez également saisir à la suite toutes les chaînes de texte, puis vous placer à la fin de chacune et insérer le saut de ligne en choisissant **XHTML** > **Break (br)**.

20 Placez-vous sur le deuxième élément de la liste à puces et saisissez `Elément LI`.

21 Débutez une liste de définitions en cliquant sur le bouton **Définition list**, puis saisissez `value`.

22 Choisissez **XHTML** > **List** > **Definition (dd)**, puis saisissez `valeur de réinitialisation`.

23 Répétez les étapes 15 à 19.

24 Placez-vous après Liste à puces, puis débutez une nouvelle liste à puces en cliquant sur le bouton **Bulleted list**.

25 Cette liste à puces doit également compter deux éléments : sélectionnez la première ligne, puis choisissez **Edit** > **Insert**.

26 Placez-vous sur la première ligne, puis saisissez `Elément UL`.

27 Débutez une liste de définitions en cliquant sur le bouton **Définition list**, puis saisissez `type`.

28 Choisissez **XHTML** > **List** > **Definition (dd)**.

29 Saisissez `disc : Rond plein (valeur par défaut)`, puis choisissez **XHTML** > **Break (br)**.

30 Répétez l'étape 29, en saisissant `square : Carré plein`, en insérant un saut de ligne, puis en saisissant `circle : Cercle vide`.

31 Placez-vous sur la seconde ligne, puis saisissez `Elément LI`.

32 Répétez les étapes 27 à 30.

La fenêtre de l'éditeur affiche directement les listes imbriquées telles que vous les aviez vues précédemment.

Figure 4.9 : Page Listes imbriquées dans l'éditeur Amaya

Aucune balise n'a été saisie « en dur ». Mais quel est l'aspect du code sous-jacent ? Pour le voir, choisissez **View** > **Show source**. Un extrait de ce code est présenté dans le listing suivant.

Figure 4.10 : Code source de la page Listes imbriquées

Remarquez la présence de numéros de lignes (servant à faciliter l'identification des erreurs), l'instruction `!doctype` et l'élément `meta`

qui concerne le générateur, automatiquement insérés. Bien que nous ayons choisi de convertir le document en HTML 4.01, les balises figurent en minuscules, conformément à la spécification XHTML 1.0. Le code est agréablement indenté et agrémenté de couleurs pour mieux identifier les éléments et leurs attributs. Remarquez en outre la présence de caractères « curieux », à la place des caractères accentués (par exemple, `ordonnée` est codé `ordonnée`). Nous reviendrons bientôt sur ce point.

Listing 4-2 : Extrait du code du fichier listesimbriquées2.html

```
<h1>Arguments et valeurs possibles dans des listes</h1>
<ol>
  <li>Liste ordonn&#xe9;e
    <ul>
      <li>El&#xe9;ment OL
        <dl>
          <dt>start</dt>
            <dd>valeur de d&#xe9;but de la num&#xe9;rotation</dd>
          <dt>type</dt>
            <dd>1,Chiffres arabes (valeur par d&#xe9;faut)<br>
              a : Lettres minuscules<br>
              A : Lettres majuscules<br>
              I : Chiffres romains en minuscules<br>
              I : Chiffres romains en majuscules.</dd>
        </dl>
      </li>
      <li>El&#xe9;ment LI
        <dl>
          <dt>value</dt>
            <dd>valeur de r&#xe9;initialisation de la
✂ num&#xe9;rotation.</dd>
          <dt>type</dt>
            <dd>1,Chiffres arabes (valeur par d&#xe9;faut)<br>
              a : Lettres minuscules<br>
              A : Lettres majuscules<br>
              I : Chiffres romains en minuscules<br>
              I : Chiffres romains en majuscules.
            </dd>
        </dl>
       </li>
    </ul>
  </li>
  <li>Liste &#xe0; puces
    <ul>
      <li>El&#xe9;ment UL
        <dl>
          <dt>type</dt>
            <dd>disc : Rond plein (valeur par d&#xe9;faut)<br>
              square : Carr&#xe9; plein<br>
              circle : Cercle vide </dd>
```

```
          </dl>
        </li>
        <li>El&#xe9;ment LI
          <dl>
            <dt>type</dt>
              <dd>disc : Rond plein (valeur par d&#xe9;faut)<br>
                square : Carr&#xe9; plein<br>
                circle : Cercle vide
              </dd>
          </dl>
        </li>
      </ul>
    </li>
  </ol>
```

L'extrait concernant uniquement les listes est un peu plus long : les balises de fermeture occupent une ligne à elles seules, contrairement au listing précédent. Remarquez en outre que les listes sont réellement imbriquées dans les éléments LI concernés : en termes de structure, cette construction est bien plus propre. Pour vous en persuader, examinez la structure du document en choisissant **View** > **Show structure**.

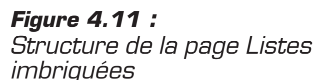


Figure 4.11 :
Structure de la page Listes imbriquées

Est-il réellement plus simple de travailler avec un éditeur dédié que directement avec le Bloc-Notes ? Pour le moment, cela n'est pas probant, mais reconnaissons que cet exemple est particulièrement « tordu » !

Les listes sont largement employées dans les pages Web. Pour vous en persuader, examinez la page de la version française de la spécification HTML 4.01 et le code source correspondant.

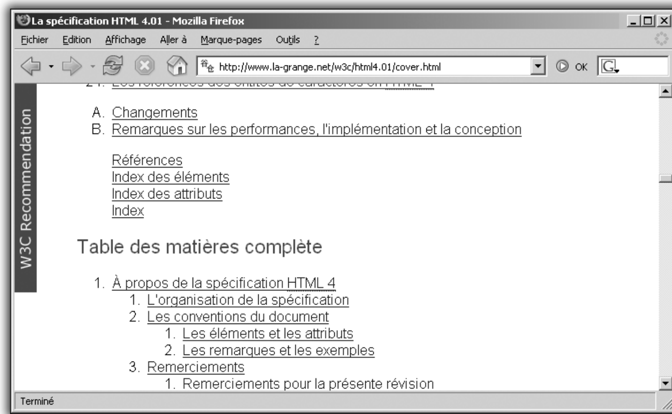


Figure 4.12 : Exemple de listes

Vous y voyez des listes à puces et numérotées imbriquées, employées pour créer une table des matières.

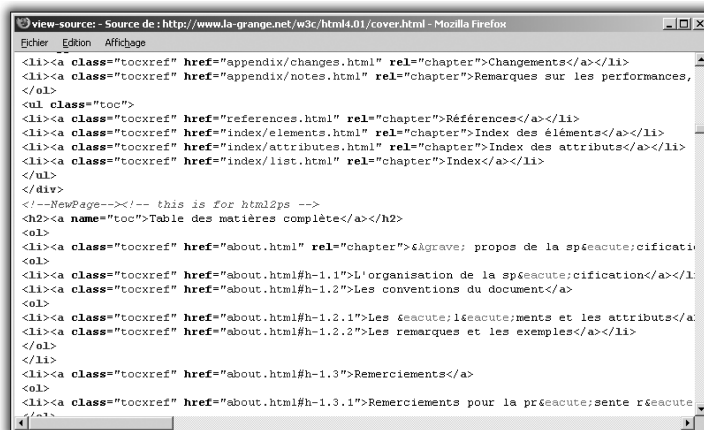


Figure 4.13 : Exemple de listes avec le code source correspondant

4.2. Tableaux

Les tableaux (souvent nommés « tables » en HTML) permettent de présenter des données (texte, texte préformaté, images, liens, formulaires, champs de formulaires, autres tables, etc.) organisées en lignes et en colonnes. Ils constituent désormais la méthode favorite d'affichage de données et de présentation des pages Web. Alors qu'ils étaient initialement prévus pour les seules données, ils ont rapidement conquis le statut d'outils de présentation, autorisant les auteurs à concevoir des pages ressemblant à des documents imprimés, dotées de barres latérales, de colonnes et d'espaces vierges.

Les tableaux ne devraient pas toutefois représenter simplement un moyen de disposer le contenu d'un document : cela peut entraîner des problèmes de restitution sur les médias non visuels. En outre, employés avec des images, ces tableaux peuvent forcer l'utilisateur à effectuer un défilement horizontal pour voir un tableau conçu sur un système offrant une surface d'affichage de taille supérieure. Afin de minimiser ces problèmes, il est recommandé d'employer des feuilles de style pour le contrôle de la disposition plutôt que des tableaux.

Les cellules d'un tableau peuvent contenir soit des informations « de rubrique » (élément `TH`), soit des informations « de données » (élément `TD`). Les cellules peuvent occuper plusieurs lignes et colonnes. Le modèle du tableau HTML 4 permet aux auteurs l'étiquetage de chaque cellule, de sorte que les agents utilisateurs non visuels peuvent communiquer plus facilement à l'utilisateur les indications de rubrique concernant la cellule. Ce mécanisme apporte non seulement une grande assistance aux personnes qui ont des déficiences visuelles, mais donne aussi aux navigateurs sans fil multimodes, avec des capacités d'affichage limitées (par exemple, les téléavertisseurs et les téléphones utilisant le Web), la possibilité de gérer les tableaux.

Contrairement à ce qui a pu se passer au début, la plupart des navigateurs prennent désormais parfaitement en charge les tableaux (même si certains interpréteurs texte-parole peuvent encore éprouver quelques difficultés à les comprendre). Compte tenu de leur facilité de mise en œuvre, il serait dommage de se priver de ces précieux outils. Attention toutefois, si leur mise en œuvre initiale est très simple, leur puissance permet d'atteindre un degré de sophistication élevé assorti d'une rare complexité !

C'est probablement pour les tableaux que le recours à un éditeur HTML dédié est le plus précieux, comme nous le verrons vers la fin du chapitre. Mais, comme nous l'avons déjà dit en d'autres cas, la meilleure façon de bien les comprendre consiste à en créer d'abord quelques-uns de toutes pièces à l'aide d'un simple éditeur de texte : ce à quoi nous allons nous attacher maintenant.

Création d'un tableau simple

La structure fondamentale d'un tableau est constituée de l'élément parent `TABLE`. Celui-ci contient tous les autres éléments qui spécifient la légende, les rangées, le contenu et la mise en forme. Ses principaux éléments enfants sont `TR` (ligne) et `TD` (cellule d'une ligne, assimilable à une colonne).

Nous allons commencer par placer sur la page *Ma famille* du site un tableau très simple, qui a pour but de présenter une famille pour le moment réduite aux deux parents (nous avons bien le temps d'avoir des enfants !).

1 Ouvrez le Bloc-Notes s'il ne l'est pas déjà, puis le fichier *famille.html* créé au cours du chapitre précédent.

2 Comme d'habitude, modifiez le numéro de version :

```
<META name="version" content="1.4.1">
```

3 Supprimez la ligne du titre de niveau 2 (`H2`), puis placez-vous après la balise `<P>` et remplacez le contenu actuel par :

```
<P>Présentation de ma famille</P>
```

4 Placez-vous après la balise `</P>` et appuyez sur [↵] pour créer une nouvelle ligne. Sur cette ligne, saisissez `<TABLE>`. Comme conseillé habituellement, fermez immédiatement l'élément en appuyant sur [↵] et en saisissant la balise fermante du tableau, `</TABLE>`.

L'astuce avec les tableaux HTML consiste à réfléchir en termes de lignes, et non de colonnes, contrairement à ce qui est généralement pratiqué dans un tableau de feuille de calcul ou de traitement de texte.

5 Placez-vous après la balise ouvrante `<TABLE>`, appuyez sur [↵], ouvrez un élément ligne avec `<TR>`, appuyez sur [↵] et fermez l'élément par `</TR>`.

6 Placez-vous après la balise ouvrante <TR>, appuyez sur ↵, ouvrez un élément colonne avec <TD>, saisissez `Qualité`, fermez l'élément par </TD>, puis appuyez sur ↵.

7 Saisissez <TD>Prénom</TD>, puis appuyez sur ↵.

8 Saisissez <TD>Age</TD>, puis appuyez sur ↵.

9 Saisissez <TD>Taille</TD>, puis appuyez sur ↵.

10 Placez-vous après la balise fermante </TR>, appuyez sur ↵, puis ouvrez un nouvel élément ligne avec <TD>. <TR>, appuyez sur ↵ et fermez l'élément par </TR>.

11 Placez-vous après la balise ouvrante <TR>, appuyez sur ↵, saisissez <TD>Père</TD>, puis appuyez sur ↵.

12 Saisissez <TD>Jean</TD>, puis appuyez sur ↵.

13 Saisissez <TD>30</TD>, puis appuyez sur ↵.

14 Saisissez <TD>1,83</TD>, puis appuyez sur ↵.

15 Placez-vous après la balise fermante </TR>, appuyez sur ↵, puis ouvrez un nouvel élément ligne avec <TR>, appuyez sur ↵ et fermez l'élément par </TR>.

16 Placez-vous après la balise ouvrante <TR>, appuyez sur ↵, saisissez <TD>Mère</TD>, puis appuyez sur ↵.

17 Saisissez <TD>Martine</TD>, puis appuyez sur ↵.

18 Saisissez <TD>28</TD>, puis appuyez sur ↵.

19 Saisissez <TD>1,63</TD>, puis appuyez sur ↵.

20 Nous agrandirons la famille plus tard. Enregistrez votre fichier sans modifier son nom (nous ne souhaitons pas de nouvelle erreur de lien !). Ne fermez ni le fichier ni le Bloc-Notes.

Figure 4.14 :
Aspect d'un tableau simple

Examinez cette page dans votre navigateur. Vous pouvez le faire en chargeant la page d'accueil et en suivant le lien vers *Ma famille*, ou en chargeant directement la page *famille.html*. Ce tableau reste d'une extrême simplicité.

Voici le code complet de cette page.

Listing 4-3 : Code de la page famille.html (version 1.4.1)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>"Ma famille"</TITLE>
<META name="author" content="mon nom">
<META name ="version" content="1.4.1">
</HEAD>
<BODY>
<H1 align="center">Ma famille</H1>
<P>Présentation de ma famille</P>
<TABLE>
<TR>
<TD>Qualité</TD>
<TD>Prénom</TD>
<TD>Age</TD>
<TD>Taille</TD>
</TR>
<TR>
<TD>Père</TD>
<TD>Jean</TD>
<TD>30</TD>
<TD>1,83</TD>
</TR>
<TR>
<TD>Mère</TD>
<TD>Martine</TD>
<TD>28</TD>
<TD>1,63</TD>
</TR>
</TABLE>
<P align="center">
<A href="mailto:votre_nom@votre_FAI">Ecrivez-moi !</A>.
</P>
<P align="center">
<A href="pageacc1_4_1.html">Retour vers la page d'accueil</A>
</P>
</BODY>
</HTML>
```

Les balises fermantes des éléments TR et TD peuvent être omises, mais cela est déconseillé, ne serait-ce que par souci de compatibilité avec XHTML 1.0.

En-tête de colonne : élément TH

C'est un tableau très simple, de trois lignes et quatre colonnes. Comment l'améliorer ? Tout d'abord, il serait mieux de mettre en évidence les en-têtes des colonnes. Cela ne pose aucun problème, grâce à un élément prévu à cet effet, TH.

- Placez-vous après l'ouverture du premier élément ligne (TR) et remplacez tous les noms d'élément TD par TH, comme dans :

```
<TH>Qualité</TH>
```

La balise fermante de l'élément TH peut être omise, mais cela reste déconseillé.



Les éléments TH

Les agents utilisateurs visuels restituent typiquement le contenu des éléments TH centré verticalement et horizontalement, et en caractères gras.

Légende de tableau : élément CAPTION

Il serait également intéressant de doter le tableau d'une légende, plutôt que d'avoir recours à un élément P qui « casse » un peu la structure de la page. Tel est le but de l'élément CAPTION.

1. Supprimez la ligne <P>Présentation de ma famille</P>.
2. Placez-vous après l'ouverture de l'élément tableau (TR), appuyez sur [↵] et saisissez :

```
<CAPTION>Présentation de la famille</CAPTION>
```

L'élément CAPTION fournit une brève description du but du tableau. Il doit impérativement être unique et situé immédiatement après la balise ouvrante de l'élément TABLE. Vous pouvez également fournir une description plus longue, grâce à l'attribut summary de l'élément TABLE, au bénéfice des personnes employant des agents utilisateurs de type convertisseur texte-parole ou génération de braille. Nous reviendrons sur

cet attribut, ainsi que sur les autres attributs applicables aux éléments enfants d'un tableau et qui permettent d'améliorer son rendu pour les agents utilisateurs non visuels dans le Chapitre 10, traitant de l'accessibilité. Servez-vous cependant tout de suite de l'attribut `summary` :

3. Placez-vous dans la balise d'ouverture de l'élément `TABLE` et saisissez :

```
<TABLE summary="Ce tableau présente les membres de ma
famille en donnant leur titre, leur prénom, leur âge
et leur taille">
```

Cet élément est particulièrement important pour les tableaux dépourvus de légende (`CAPTION`).

Contrôle de l'alignement d'un tableau

Vous avez déjà rencontré l'attribut `align`, disponible pour à peu près tous les éléments bloc. `TABLE` étant un élément bloc, en employant `align="center"` dans cet élément, vous centrez le tableau dans la page. Les valeurs possibles pour `align` sont `center`, `left` ou `right`.

1. Modifiez la ligne de la balise d'ouverture de l'élément `TABLE` pour centrer le tableau :

```
<TABLE align="center" summary="Ce tableau présente
les membres de ma famille en donnant leur titre,
leur prénom, leur âge et leur taille">
```

2. Enregistrez votre fichier, sans modifier son nom. Ne fermez ni le fichier ni le Bloc-Notes.

Examinez cette page dans votre navigateur. Le tableau est plus intéressant.

Figure 4.15 :
Aspect du tableau centré

Vous pouvez également employer cet attribut dans une ligne, pour ne centrer que les contenus des cellules de cette ligne, ou dans n'importe laquelle des cellules individuelles, pour ne centrer que les données de cette cellule. Souvenez-vous cependant que l'emploi de l'attribut `align` est désormais est désormais déconseillé : mieux vaut recourir aux feuilles de style.

Bordures et règles de tableau : attributs border, frame et rules

Un tableau digne de ce nom possède généralement des lignes de séparation. Vous pouvez contrôler celles-ci à l'aide de plusieurs attributs facultatifs de l'élément `TABLE` : `border`, `frame` et `rules`.

Commençons par l'attribut `border`, dont la valeur exprimée en pixels définit la largeur du cadre extérieur.

1. Modifiez comme suit la ligne de la balise d'ouverture de l'élément `TABLE` :

```
<TABLE border="2" align="center" summary="Ce tableau
présente les membres de ma famille en donnant leur
titre, leur prénom, leur âge et leur taille">
```

2. Enregistrez votre fichier, sans modifier son nom. Ne fermez ni le fichier ni le Bloc-Notes.

Examinez le tableau dans un navigateur : son aspect est nettement plus sympathique.

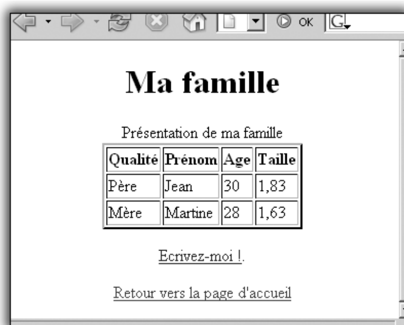


Figure 4.16 :
Aspect du tableau amélioré

Vous pouvez cependant aller bien plus loin, en spécifiant quels côtés du cadre entourant la table doivent être visibles (grâce à l'attribut `frame`),

ainsi que les règles devant être affichées (grâce à l'attribut `rules`). Les valeurs possibles pour ces deux attributs sont présentées dans le tableau de la page 5. Modifiez votre tableau pour afficher une bordure de 5 pixels à gauche et à droite du tableau, des règles étant dessinées uniquement entre les colonnes :

3 Modifiez comme suit la ligne de la balise d'ouverture de l'élément `TABLE` :

```
<TABLE border="5" frame="vsides" rules="cols" align="center"
summary="Ce tableau présente les membres de ma famille en
donnant leur titre, leur prénom, leur âge et leur taille">
```

4 Enregistrez votre fichier, sans modifier son nom. Ne fermez ni le fichier ni le Bloc-Notes.

Examinez le résultat obtenu dans un navigateur : son aspect devrait ressembler à ce qui est présenté dans la figure suivante..

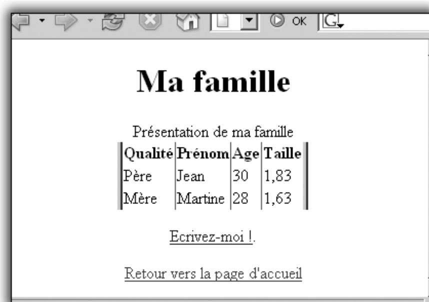


Figure 4.17 : *Aspect du nouveau tableau avec Firefox*

Le tableau suivant présente les valeurs possibles des attributs `frame`, `rules` et `border` ainsi que leur signification.

Tableau 4.2 : *Attributs de bordure et de règles de l'élément TABLE*

Attribut	Valeur	Signification
`Frame`	`Void`	Aucune bordure (valeur par défaut)
	`Above`	Haut uniquement
	`Below`	Bas uniquement
	`Hsides`	Haut et bas uniquement
	`Vsides`	Droite et gauche uniquement
	`Lhs`	Côté gauche uniquement

Tableau 4.2 : Attributs de bordure et de règles de l’élément TABLE		
Attribut	**Valeur**	**Signification**
	Rhs	Côté droit uniquement
	box ou border	Les quatre côtés
Rules	None	Aucune (valeur par défaut)
	Groups	Uniquement entre les groupes de lignes (THEAD, TFOOT et TBODY) et les groupes de colonnes (COLGROUP et COL)
	Rows	Uniquement entre les lignes
	Cols	Uniquement entre les colonnes
	All	Entre toutes les lignes et colonnes
Border	N	Bordure de n pixels. Pas de bordure si n = 0.

Pour des raisons de compatibilité descendante, les conventions suivantes doivent être respectées par les agents utilisateurs :

- La définition `border="0"` signifie que `frame="void"` et que `rules="none"` (sauf si spécifié autrement).
- Toute autre valeur de l’attribut `border` signifie inversement que `frame="border"` et que `rules="all"`.
- La présence de l’attribut `border` sans valeur dans la balise ouvrante de l’élément `TABLE` équivaut à `frame="border"` et à `rules="all"`, avec une certaine valeur par défaut non nulle pour l’attribut `border`.
- Autrement dit, les définitions `<TABLE border="2">` et `<TABLE border="2" frame="border" rules="all">` sont équivalentes, comme `<TABLE border>` et `<TABLE frame="border" rules="all">`.



Variations selon les navigateurs

Le rendu d’un tableau peut différer légèrement selon le navigateur employé. Examinez les copies d’écran suivantes, correspondant à l’état actuel de notre page *famille* vue sous Internet Explorer





Figure 4.18 :
Aspects du nouveau tableau avec Internet Explorer

et Amaya,

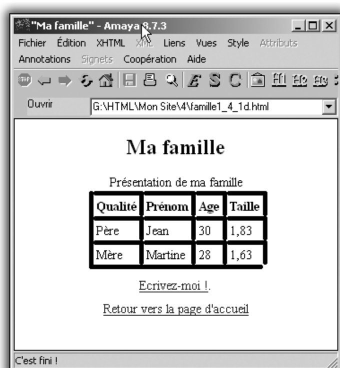


Figure 4.19 :
Aspects du nouveau tableau avec Amaya

et comparez-les à la celle obtenue avec Firefox précédemment obtenu.

Si Firefox et Internet Explorer sont très proches (l'intervalle entre le titre et le tableau est plus faible avec Firefox, tandis que l'effet « relief » est plus prononcé), Amaya affiche une unique ligne de séparation.

Contrôle de la taille et de la structure des cellules

Reprenons un peu de terminologie : un tableau est composé de *lignes* (éléments `TR`) et de *colonnes*. L'intersection d'une ligne et d'une colonne est une *cellule*, le plus petit élément de données d'un tableau. Une cellule individuelle ne devrait contenir qu'une information

(toujours la structure avant tout !), de type en-tête (élément TH) ou donnée (élément TD). Dans ce qui précède, vous avez construit votre premier tableau, composé de trois lignes (dont la ligne d'en-têtes) et de quatre colonnes, soit un total de douze cellules individuelles : le tableau contient effectivement trois éléments ligne TR et douze éléments cellule, dont quatre éléments TH d'en-tête et huit éléments TD de données.

Une cellule peut recevoir une mise en forme particulière en HTML à l'aide d'attributs spéciaux.

Espacement : attributs cellspacing et cellpadding

Il est ainsi possible de modifier l'espacement entre les cellules, tout comme le décalage entre le contenu de la cellule et la bordure, ce qui porte le nom un peu inattendu de « remplissage » *(padding)*. Pour ce faire, vous modifiez les attributs `cellspacing` et/ou `cellpadding`.

L'attribut `cellspacing` détermine la largeur de la partie blanche de la bordure. Autrement dit, il quantifie l'espace qui doit être laissé par l'agent utilisateur entre le côté gauche du tableau et le côté gauche de la colonne située à l'extrême gauche, le haut du tableau et le haut de la rangée supérieure, et ainsi de suite pour le côté droit et le bas du tableau. L'attribut spécifie également l'espacement entre les cellules.

Notre tableau étant de très petite taille, vous allez l'aérer un peu.

1. Modifiez comme suit la ligne de la balise d'ouverture de l'élément TABLE :

```
<TABLE border="5" frame="vsides" rules="cols" align="center"
cellspacing="20" summary="Ce tableau présente les membres
de ma famille en donnant leur titre, leur prénom, leur âge
et leur taille">
```

2. Enregistrez votre fichier, sans modifier son nom, puis examinez le résultat dans votre navigateur (voir Figure 4.20).

À moins que cela ne ressemble à une des figures qui suivent (voir Figure 4.21, 4.22).

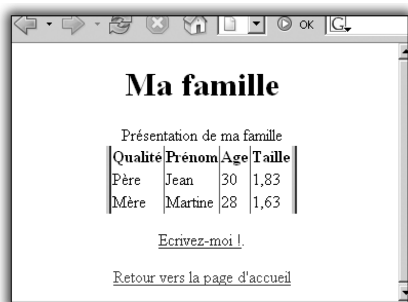


Figure 4.20 :
Emploi de l'attribut cellspacing, vu dans Firefox.

Qualité	Prénom	Age	Taille
Père	Jean	30	1,83
Mère	Martine	28	1,63

Figure 4.21 :
Emploi de l'attribut cellspacing, vu dans Internet Explorer

Qualité	Prénom	Age	Taille
Père	Jean	30	1,83
Mère	Martine	28	1,63

Figure 4.22 :
Emploi de l'attribut cellspacing, vu dans Amaya.

Les différences entre navigateurs sont ici encore plus flagrantes : Firefox semble incapable de tenir compte de la présence de l'attribut `cellspacing`, contrairement à ses homologues. Plus vous compliquez la présentation d'un tableau en combinant les attributs et plus cela est susceptible de se produire.

3 Modifiez à nouveau la ligne de la balise d'ouverture de l'élément `TABLE` :

```
<TABLE border="2" align="center" cellspacing="20" summary=
"Ce tableau présente les membres de ma famille en donnant
leur titre, leur prénom, leur âge et leur taille">
```

L'aspect dans les trois navigateurs, bien qu'encore différent, est plus satisfaisant.

Figure 4.23 :
Page modifiée avec attribut cellspacing, vu dans Firefox

Figure 4.24 :
Page modifiée avec attribut cellspacing, vu dans Internet Explorer

Figure 4.25 :
Page modifiée avec attribut cellspacing, vu dans Amaya.

Un second attribut, `cellpadding`, correspond à la largeur, également exprimée en pixels, de l'espace compris entre le contenu de la cellule et la bordure du tableau ou la règle. La valeur de cet attribut peut être exprimée en pixels : les quatre marges se trouvent alors à cette distance du contenu. Si la valeur de l'attribut est exprimée en pourcentage, les marges supérieure et inférieure devraient se trouver à égale distance du contenu, en fonction d'un pourcentage de l'espacement vertical disponible, tandis que les marges gauche et droite devraient se trouver à égale distance du contenu, en fonction d'un pourcentage de l'espacement horizontal disponible.

Ces deux attributs contrôlent donc conjointement l'espacement entre et à l'intérieur des cellules.



Bordure invisible

En attribuant au tableau une bordure invisible (`border=0`), le contenu de la cellule semble « flotter », séparé de toutes les autres informations.



> C'est souvent intéressant, puisque HTML reste globalement peu adapté aux espaces blancs.

1 Modifiez comme suit la ligne de la balise d'ouverture de l'élément `TABLE` :

```
<TABLE border="2" align="center" cellspacing="20"
cellpadding="30%" summary="Ce tableau présente les membres
de ma famille en donnant leur titre, leur prénom, leur
âge et leur taille">
```

2 Enregistrez votre fichier, sans modifier son nom, puis examinez le résultat dans votre navigateur. Vous devriez voir quelque chose d'équivalent à ce qui est présenté dans la figure suivante.

Présentation de ma famille

Qualité	Prénom	Age	Taille
Père	Jean	30	1,83
Mère	Martine	28	1,63

Figure 4.26 : *Emploi de l'attribut cellpadding*

Ici, l'attribut `cellspacing` spécifie que les cellules devraient être séparées de vingt pixels, les unes par rapport aux autres ainsi que par rapport au cadre du tableau. L'attribut `cellpadding` stipule que la marge supérieure et la marge inférieure de la cellule seront chacune séparées du contenu de la cellule de 15 % de l'espacement vertical disponible (soit 30 % au total). De même, la marge de gauche et la marge de droite de la cellule seront chacune séparées du contenu de la cellule de 15 % de l'espacement horizontal disponible (soit 30 % au total).

Alignement : attributs align et valign

Vous pouvez également modifier l'alignement d'une cellule. Pour modifier l'alignement horizontal, vous recourez à l'attribut `align`, déjà

rencontré, dans un élément `TD` ou `TH`. L'alignement vertical s'ajuste à l'aide de l'attribut `valign`. `valign`, qui peut prendre les valeurs `top` (haut), `bottom` (bas), `middle` (milieu, la valeur par défaut) et `baseline` (première ligne). Cette dernière valeur aligne le contenu de toutes les cellules selon la première ligne de texte, mais pas forcément en haut ni en alignant la dernière ligne.

1 Modifiez comme suit les éléments `TD` du second élément `TR` (le premier contenant les en-têtes, soit des éléments `TH`) :

```
<TD valign="top">Père</TD>
<TD valign="top">Jean</TD>
<TD align="right" valign="top">30</TD>
<TD align="right" valign="top">1,83</TD>
```

2 Procédez de même pour les éléments `TD` du dernier élément `TR` :

```
<TD valign="top">Mère</TD>
<TD valign="top">Martine</TD>
<TD align="right" valign="top">28</TD>
<TD align="right" valign="top">1,63</TD>
```

3 Enregistrez votre fichier, sans modifier son nom, puis examinez le résultat dans votre navigateur.

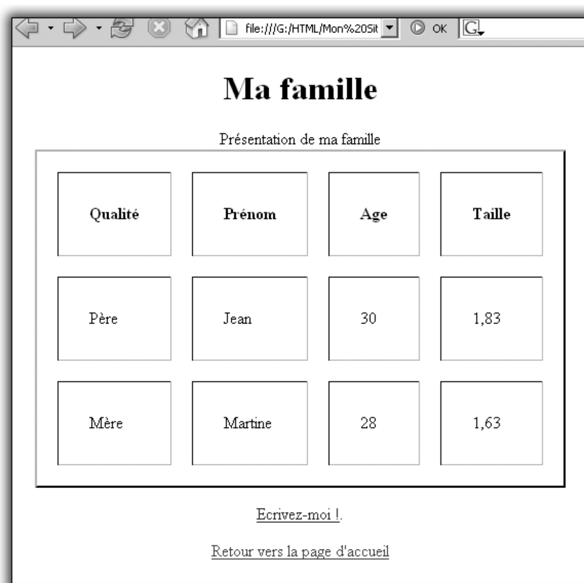


Figure 4.27 : *Contrôle de l'alignement horizontal et vertical*



Attributs déconseillés en HTML

Vous ne devriez pas être étonnés que je signale une fois encore que les attributs `align` et `valign` sont à déconseiller dans le corps de la page HTML, et qu'une feuille de style devrait être employée à la place…

Largeur des colonnes : attribut width

La largeur des colonnes peut être spécifiée à l'aide de l'argument `width` de trois façons différentes :

- **Fixe** : une largeur fixe est exprimée en pixels (par exemple, `width="30"`). Spécifier une largeur fixe permet une restitution progressive.



Conflit d'attributs

Lorsqu'un tableau ou une colonne possède une largeur fixe, les attributs `cellspacing` et `cellpadding` peuvent nécessiter plus d'espace que celui qui est attribué. En cas de conflit, les agents utilisateurs peuvent (sans être obligés de le faire) donner à ces attributs la priorité sur l'attribut de largeur `width`. Mieux vaut donc éviter de se trouver dans cette situation.

- **Pourcentage** : une largeur exprimée en pourcentage (par exemple, `width="20%"`) se fonde sur le pourcentage d'espace horizontal disponible pour le tableau (entre les marges courantes gauche et droite). Remarquez que cet espace ne dépend pas du tableau en lui-même : une spécification en pourcentage permet donc une restitution progressive.
- **Proportionnelle** : une valeur proportionnelle (par exemple, `width="3*"`) se rapporte aux portions d'espace horizontal nécessaire au tableau. Si le tableau possède une largeur fixe (*via* l'attribut `width` de l'élément `TABLE`), les agents utilisateurs peuvent le restituer de façon progressive, comme pour des colonnes proportionnelles. En revanche, si la largeur du tableau n'est pas fixe, l'agent utilisateur doit attendre d'avoir reçu toutes les données avant de pouvoir déterminer l'espace horizontal nécessaire au tableau. Cet espace sera alors seulement alloué aux colonnes proportionnelles.

Comme cela a été mentionné, l'attribut `width` peut également être employé sur l'élément `TABLE` pour définir la largeur totale d'un tableau. Comme pour une colonne, cette largeur peut être exprimée de façon fixe, en pourcentage ou de façon proportionnelle.

Modifiez de nouveau votre tableau :

1 Modifiez comme suit la ligne de la balise d'ouverture de l'élément `TABLE`, pour créer un tableau occupant 80 % de l'espace libre disponible à l'écran :

```
<TABLE border="2" align="center" cellspacing="20"
cellpadding="30%" width="80%" summary="Ce tableau
présente les membres de ma famille en donnant leur
titre, leur prénom, leur âge et leur taille">
```

2 Modifiez comme suit les éléments `TH` du premier élément `TR` (celui qui contient les en-têtes) :

```
<TH width="30%">Qualité</TH>
<TH width="30%">Prénom</TH>
<TH width="20%">Age</TH>
<TH width="20%">Taille</TH>
```

Remarquez que le total des quatre pourcentages est bien égal à 100 %.

3 Enregistrez votre fichier, sans modifier son nom, puis examinez le résultat dans votre navigateur.

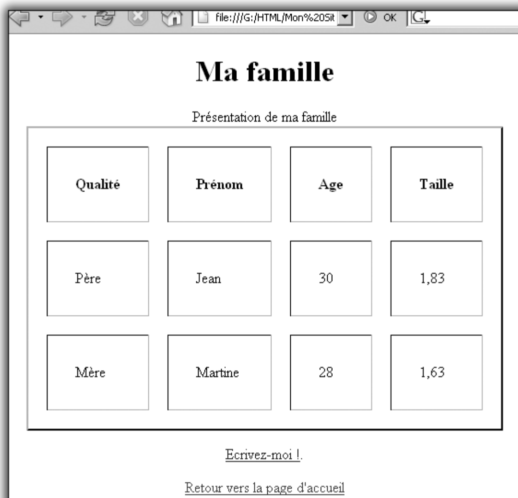


Figure 4.28 : *Ajustement de la largeur du tableau et des cellules*

Cette figure mérite que vous vous y attardiez un peu plus longtemps. Tout d'abord, remarquez que la valeur par défaut de l'attribut d'alignement horizontal `align` est « centré » (`align="center"`) pour un élément `TH`, alors qu'il est « à gauche » (`align="left"`) pour un élément `TD`. En outre, les choix d'alignement sont bien mieux mis en évidence que précédemment. Remarquez également qu'il a suffi de modifier la largeur des cellules de la première ligne : ces spécifications ont été héritées par les lignes suivantes.

Mais que se passerait-il si vous spécifiez des largeurs différentes pour les cellules de la seconde ligne ? Autant essayer…

4 Modifiez comme suit les éléments `TD` du second élément `TR` (le premier contenant les en-têtes, soit des éléments `TH`) :

```
<TD valign="top" width="20%">Père</TD>
<TD valign="top" width="20%">Jean</TD>
<TD align="right" valign="top" width="30%">30</TD>
<TD align="right" valign="top" width="25%">1,83</TD>
```

Pour corser les choses, le total des pourcentages est même cette fois différent de 100 %…

5 Enregistrez votre fichier, mais cette fois-ci sous le nom *FamilleTest.html*, puis examinez le résultat dans votre navigateur.



Figure 4.29 : *Ajustement incorrect de la largeur du tableau et des cellules*

Manifestement, l'agent utilisateur a été déconcerté : il a retenu la plus forte des valeurs de largeur spécifiées pour les trois premières colonnes (30 % pour les deux premières colonnes, à cause des largeurs spécifiées pour les deux premières cellules de la première ligne, puis 30 % pour la

troisième colonne, à cause de la troisième cellule de la deuxième ligne) et n'a accordé à la dernière colonne que le reste de l'espace disponible, soit 10 %.

Gardez toutefois à l'esprit que, même si l'emploi de l'attribut `width` n'est pas déconseillé, mieux vaut spécifier la largeur des tableaux à l'aide de feuilles de style.

Fusion de cellules : attributs rowspan et colspan

Dans un tableau, il est souvent utile de fusionner des cellules, que cela soit horizontalement (sur plusieurs colonnes) ou verticalement (sur plusieurs lignes), voire les deux. L'attribut `rowspan` réalise une fusion verticale tandis que l'attribut `colspan` opère une fusion horizontale.

L'attribut `rowspan` spécifie le nombre de colonnes couvertes par la cellule courante. Sa valeur par défaut est `"1"`, la valeur `"0"` signifiant que la cellule couvre toutes les colonnes, depuis la colonne courante jusqu'à la dernière colonne de la section de table (voir plus loin les éléments `THEAD`, `TBODY` et `TFOOT`) dans laquelle la cellule est définie.

L'attribut `colspan` spécifie le nombre de colonnes couvertes par la cellule courante. Sa valeur par défaut est `"1"`, la valeur `"0"` signifiant que la cellule couvre toutes les colonnes, à partir de la colonne courante jusqu'à la dernière colonne du groupe de colonnes (voir plus loin l'élément `COLGROUP`) dans lequel la cellule est définie.

Création d'un tableau à cellules fusionnées à l'aide d'un éditeur de texte

Vous allez ici mettre en œuvre ces nouveaux attributs en modifiant la page secondaire *Mes régions* du site Web, à l'aide d'un éditeur de texte.

1 Ouvrez le Bloc-Notes s'il ne l'est pas déjà, puis le fichier *region.html* (attention, si vous n'avez pas fermé précédemment le Bloc-Notes, le fichier actuel est *familleTest.html*).

2 Comme d'habitude, modifiez le numéro de version :

```
<META name ="version" content="1.4.2">
```

3 Supprimez la ligne du titre de niveau 2 (`H2`), puis le paragraphe de l'élément `P`. Créez à sa place un nouveau tableau centré, avec sa légende (élément `CAPTION`) et son attribut `summary`) :

```
<TABLE border="2" align="center" summary=
"Présentation de ma région">
<CAPTION>Ma région</CAPTION>
</TABLE>
```

4 Vous allez créer ici un tableau un peu complexe, employant à la fois des fusions de lignes et de colonnes, dont l'aspect théorique est représenté dans la figure suivante.

<table>
<caption>Tableau 4.3 : Aspect du tableau à créer</caption>
<tr><th></th><th>Colonne 1</th><th>Colonne 2</th><th>Colonne 3</th><th>Colonne 4</th></tr>
<tr><td>Ligne 1</td><td rowspan="4">Cette cellule occupe 1 colonne et 4 lignes. Elle contiendra par la suite une image.</td><td colspan="3">Cette cellule s'étend sur 3 colonnes.</td></tr>
<tr><td>Ligne 2</td><td>Cellule individuelle</td><td>Cellule individuelle</td><td>Cellule individuelle</td></tr>
<tr><td>Ligne 3</td><td rowspan="2">Cellule sur 2 lignes</td><td rowspan="2" colspan="2">Cellule sur 2 lignes et 2 colonnes</td></tr>
<tr><td>Ligne 4</td></tr>
</table>

Ce tableau possède au total quatre lignes et quatre colonnes. Comme cela a déjà été évoqué précédemment, un tableau HTML doit se construire ligne par ligne. Examinez la première ligne théorique : elle contient deux cellules, l'une s'étendant sur trois lignes et l'autre sur trois colonnes. Entrez le code pour créer cette première ligne.

5 Ouvrez un élément de ligne `TR` (en vous plaçant au-dessus de la balise de fin `</TABLE>` et en créant de nouvelles lignes), puis créez la première cellule en la déclarant comme s'étendant sur 4 lignes :

```
<TR>
<TD rowspan="4">...Une image sera placée ici...</TD>
</TR>
</TABLE>
```

6 Créez ensuite la seconde cellule, qui doit s'étendre sur trois colonnes :

```
<TR>
<TD rowspan="4">...Une image sera placée ici...</TD>
<TD colspan="3">Ma région est célèbre pour ses forêts,
ses montagnes et ses lacs.</TD>
</TR>
```

Tableau 4.4 : Cellules créées à la fin de l'étape 6				
	Colonne 1	**Colonne 2**	**Colonne 3**	**Colonne 4**
Ligne 1	…Une image sera placée ici…	Ma région est célèbre pour ses forêts, ses montagnes et ses lacs.		
Ligne 2		Cellule individuelle	Cellule individuelle	Cellule individuelle
Ligne 3		Cellule sur 2 lignes	Cellule sur 2 lignes et 2 colonnes	
Ligne 4				

7 Passons à la seconde ligne. Ouvrez un nouvel élément `TR`, sous la ligne précédente, puis créez ses cellules. Vous n'en avez que trois à créer, puisque la première cellule de cette ligne est sous-entendue par l'extension de la première cellule de la première ligne.

```
<TR>
<TD>Les forêts</TD>
<TD>Les montagnes</TD>
<TD>Les lacs</TD>
</TR>
</TABLE>
```

Tableau 4.5 : Cellules crées à la fin de l'étape 7				
	Colonne 1	**Colonne 2**	**Colonne 3**	**Colonne 4**
Ligne 1	…Une image sera placée ici…	Ma région est célèbre pour ses forêts, ses montagnes et ses lacs.		
Ligne 2		Les forêts	Les montagnes	Les lacs
Ligne 3		Cellule sur 2 lignes	Cellule sur 2 lignes et 2 colonnes	
Ligne 4				

8 Occupez-vous de la troisième ligne. Créez cette fois uniquement deux éléments `TD`, l'un s'étendant sur deux lignes et l'autre occupant deux lignes et deux colonnes :

```
<TR>
<TD rowspan="2">Ici, du texte à compléter </TD>
<TD rowspan="2" colspan="2">Ici, une autre image </TD>
</TR>
</TABLE>
```

Tableau 4.6 : *Cellules créées à la fin de l'étape 8*

<table>
<tr><th></th><th>Colonne 1</th><th>Colonne 2</th><th>Colonne 3</th><th>Colonne 4</th></tr>
<tr><td>Ligne 1</td><td rowspan="4">…Une image sera placée ici…</td><td colspan="3">Ma région est célèbre pour ses forêts, ses montagnes et ses lacs.</td></tr>
<tr><td>Ligne 2</td><td>Les forêts</td><td>Les montagnes</td><td>Les lacs</td></tr>
<tr><td>Ligne 3</td><td rowspan="2">Ici, du texte à compléter.</td><td colspan="2" rowspan="2">Ici, une autre image.</td></tr>
<tr><td>Ligne 4</td></tr>
</table>

9 Et la quatrième ligne ? Comme le montre la figure précédente, elle est déjà implicitement créée : sa première cellule par la première cellule de la première ligne, la seconde par la seconde cellule de la troisième ligne et les deux dernières par la troisième cellule de la ligne précédente. Pour la bonne forme, insérez tout de même un élément `TR` vide :

```
<TR></TR>
```

10 Enregistrez le fichier, puis examinez la page dans votre navigateur:

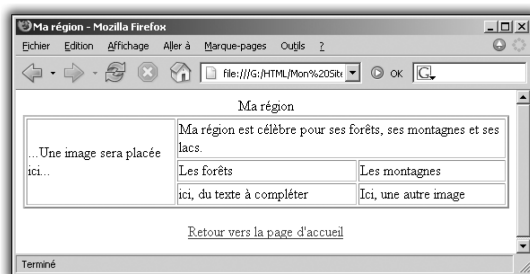


Figure 4.30 : *Aspect de la page Région dans un navigateur*

Voici le code complet de la page :

Listing 4-4 : Fichier region.html (version 1.4.2).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"><HTML>
<HTML>
  <HEAD>
    <TITLE>Ma région</TITLE>
    <META name="author" content="Fabrice Lemainque">
    <META name ="version" content="1.4.2">
  </HEAD>
  <BODY>
```

```
    <TABLE border="2" align="center" summary=
      "Présentation de ma région">
      <CAPTION>Ma région</CAPTION>
      <TR>
        <TD rowspan="4">...Une image sera placée ici...</TD>
        <TD colspan="3">Ma région est célèbre pour ses forêts,
        ses montagnes et ses lacs.</TD>
      </TR>
      <TR>
        <TD>Les forêts</TD>
        <TD>Les montagnes</TD>
        <TD>Les lacs</TD>
      </TR>
      <TR>
        <TD rowspan="2">ici, du texte à compléter </TD>
        <TD rowspan="2" colspan="2">Ici, une autre image</TD>
      </TR>
    </TABLE>
    <P align="center">
      <A href="pageacc1_4_1.html">Retour vers la page
       d'accueil</A>
    </P>
  </BODY>
</HTML>
```

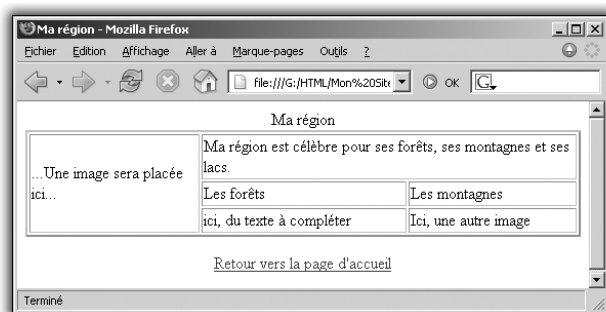


Figure 4.31 : *Aspect de la page Région dans un navigateur*

Une bonne méthode consiste à toujours procéder comme nous l'avons fait ici, c'est-à-dire en traçant un quadrillage sur une feuille de papier et en indiquant les cellules déjà créées : cela diminue fortement les risques d'erreur et les surprises désagréables susceptibles d'en résulter.

Création d'un tableau à cellules fusionnées à l'aide d'un éditeur HTML

Comme nous l'avons dit auparavant, les tableaux tel celui de cet exemple peuvent rapidement devenir très complexes : ce sont

probablement eux qui méritent le plus le recours à un éditeur HTML. Nous allons concevoir à nouveau le même tableau, mais cette fois à l'aide de Nvu.

1. Ouvrez Nvu. Bonne nouvelle, contrairement à Amaya, l'interface est localisée, si toutefois vous avez téléchargé la bonne version (nous employons ici la version 1.0PR (20050330)).
2. Saisissez `Ma région`. Cliquez sur la flèche de la zone de liste déroulante, à côté de **Corps de texte** et, choisissez dans la liste **Titre 1**.

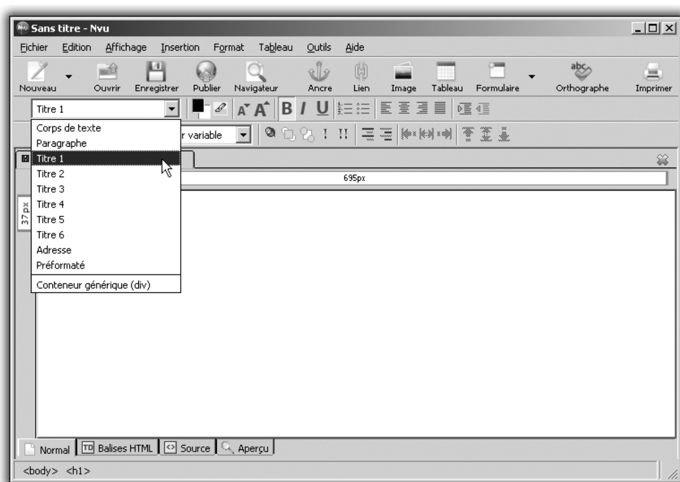


Figure 4.32 : *Création d'un titre de niveau 1*

Placez-vous à la fin du titre, puis appuyez sur ↵.

3. Cliquez sur l'icône *Tableau*. Cela affiche une nouvelle fenêtre, qui permet de définir le tableau à créer.

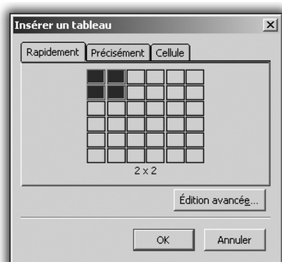


Figure 4.33 : *Boîte de dialogue Insérer un tableau*

Vous allez immédiatement modifier les attributs de votre tableau (en réalité de l'élément `TABLE`). Cliquez sur **Édition avancée**. Vous voyez que Nvu a déjà défini trois attributs avec des valeurs par défaut : `cellspacing`, `cellpading` et `border`.

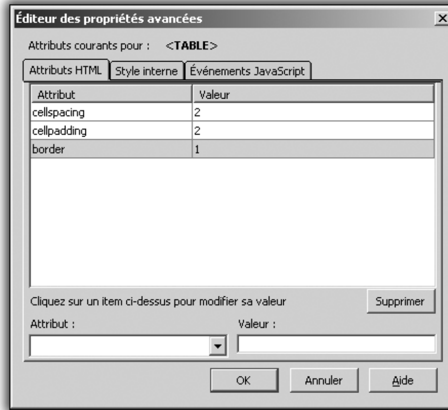


Figure 4.34 : *Éditeur des propriétés avancées*

4 Vous voulez ajouter deux attributs `summary` et `align`. Ouvrez la zone de liste déroulante *Attribut* et choisissez `summary`. Dans la zone de texte *Valeur*, saisissez `Présentation de ma région`. Ouvrez à nouveau la zone de liste déroulante *Attribut* et choisissez *align*. Ouvrez la zone de liste déroulante *Valeur* et choisissez *center*. La fenêtre **Éditeur des propriétés avancées** doit ressembler à ce qui est présenté dans la figure qui suit. Confirmez par OK pour revenir à la fenêtre **Insérer un tableau**.

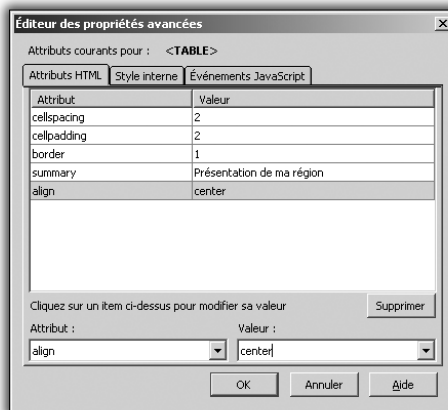


Figure 4.35 : *Fenêtre Éditeur des propriétés avancées renseignée*

5 Nous voulons un tableau de quatre lignes et quatre colonnes : sélectionnez dans cette fenêtre quatre lignes et quatre colonnes, puis cliquez sur OK.

6 Sélectionnez les cellules de la première colonne, puis choisissez dans le menu **Tableau** > **Fusionner les cellules sélectionnées** : les quatre cellules concernées fusionnent.

7 Sélectionnez les trois dernières cellules de la première ligne, puis choisissez à nouveau dans le menu **Tableau** > **Fusionner les cellules sélectionnées** : les trois cellules concernées fusionnent.

8 De la même façon, faites fusionner les secondes cellules des lignes 3 et 4, puis les deux dernières cellules des lignes 3 et 4.

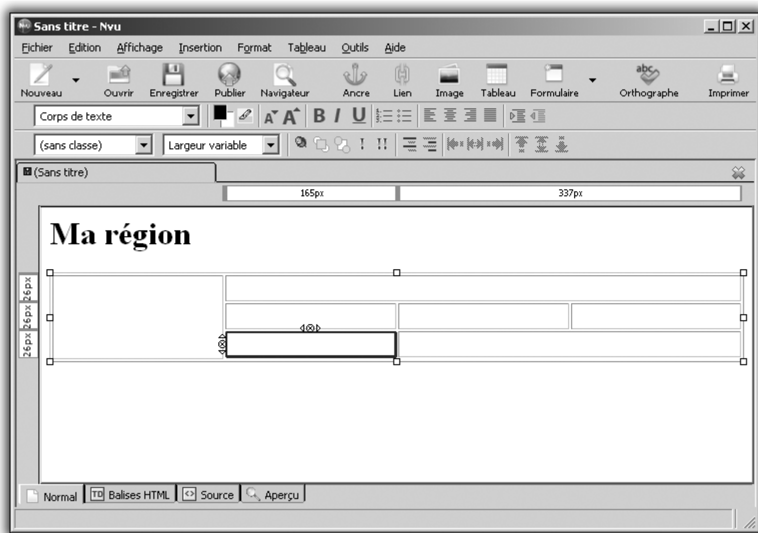


Figure 4.36 : *Aspect de la page Ma région après l'étape 8*

9 Placez-vous dans la première cellule et saisissez `...Une image sera placée ici…` Placez-vous dans la seconde cellule de la première ligne et saisissez `Ma région est célèbre pour ses forêts, ses montagnes et ses lacs.` Saisissez dans les cellules vides de la seconde ligne respectivement `Les forêts`, `Les montagnes` et `Les lacs`.

10 Sur la troisième ligne, saisissez dans la seconde cellule `Ici, du texte à compléter` et dans la troisième `Ici, une autre image`.

11 Il subsiste quelques détails à régler : placez-vous sur le titre de niveau 1 et choisissez dans la barre d'outils **Format** > **Aligner** > **Centrer**. Cela centre le titre. Ensuite, vous ne devez pas oublier d'ajouter un nom d'auteur. Cela est accompli à l'aide de l'option de menu *Outils* > *Préférences*. La page *Options* s'affiche. Cliquez sur **Paramètres de pages** et, dans la fenêtre de droite, saisissez dans le champ *Auteur* `mon nom` (où `mon nom` est votre nom).

12 Le tableau obtenu ressemble beaucoup à ce que nous avions fait précédemment, mais a été réalisé ici avec une grande facilité. Enregistrez votre travail. Choisissez **Fichier** > **Enregistrer sous**. Le programme vous demande de saisir un titre (le contenu de l'élément `TITLE`). Saisissez `Ma région` et cliquez sur OK. Dans la boîte de dialogue **Enregistrer sous**, naviguez jusqu'au dossier *Mon Site Web* et enregistrez le fichier sous le nom *region2.html*.



Figure 4.37 : *Aspect de la page terminée dans l'éditeur Nvu*

13 Il serait maintenant intéressant d'examiner le code source ainsi généré. Rien de plus simple : cliquez en bas de la fenêtre sur l'onglet **Source**. Le code source s'affiche (voir Figure 4.38).

Ce code est très semblable à celui créé manuellement. Remarquez toutefois quelques différences :

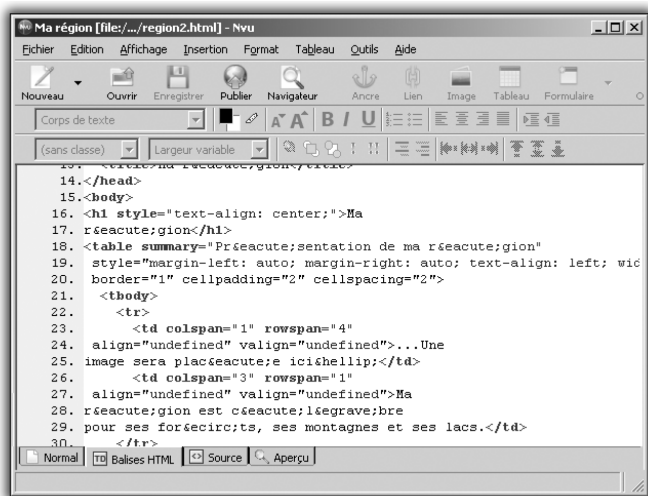


Figure 4.38 : *Code source de la page*

- Comme nous l'avons déjà vu auparavant, les caractères accentués sont représentés par des séquences « bizarres » de caractères. Nous examinerons ceux-ci dans le prochain chapitre.
- La balise `TABLE` emploie un nouvel attribut (`style= "margin-left: auto; margin-right: auto; text-align: left; width: 100%;"`). Il sera examiné en détail dans le Chapitre 8, traitant des feuilles de style.
- Les balises `TD` se voient dotées d'attributs que nous connaissons, mais possédant la valeur `"undefined"` (`<td colspan="1" rowspan="4" align="undefined" valign="undefined">`). C'est un des petits problèmes des éditeurs HTML, qui insèrent automatiquement des balises et attributs supplémentaires, ce qui alourdit le code. Lorsque HTML constate l'absence de ces attributs avec une valeur spécifiée, il considère qu'ils sont présents avec leur valeur par défaut : il est donc superflu de les ajouter, cela alourdissant inutilement le code.
- Il manque cependant l'élément `META` indiquant la version du fichier : ajoutez-la manuellement, puis enregistrez le fichier en cliquant sur **Enregistrer**.

Comme vous venez de le voir, il est bien plus simple de créer un tableau à l'aide d'un éditeur HTML comme Nvu que de le bâtir de toutes pièces dans un simple éditeur de texte, à moins de faire preuve d'une rare maîtrise du code HTML !

Cellules manquantes et cellules se recoupant

Deux problèmes sont fréquents avec les tableaux possédant des cellules fusionnées : l'oubli d'une cellule dans une ligne et le recouvrement de cellules. Pour voir ce qui se passe dans ces conditions, nous allons modifier à l'aide d'un éditeur de texte la page *région.html*.

1 Ouvrez le Bloc-Notes s'il ne l'est pas déjà, puis ouvrez le fichier *region.html*.

2 Par sécurité, enregistrez-le tout de suite sous le nom *region_erreur.html*.

3 Modifiez le titre (élément `TITLE`) :

```
<TITLE>Ma région (page avec erreurs)</TITLE>
```

4 Comme d'habitude, modifiez le numéro de version :

```
<META name ="version" content="1.4.1">
```

5 Modifiez la légende (élément `CAPTION`) :

```
<CAPTION>Ma région (tableau avec erreurs)</CAPTION>
```

6 Vous allez d'abord créer une nouvelle ligne en « oubliant » une cellule. Pour ce faire, placez-vous avant la balise de fermeture `</TABLE>`, appuyez sur ↵ puis saisissez :

```
<TR>
<TD><BR></TD>
<TD><BR></TD>
<TD><BR></TD>
</TR>
```

Une astuce a été utilisée ici : pour créer une cellule vide qui possède toutefois un écartement entre ses bordures, insérez dans celle-ci un saut de ligne (`<BR>`). Cela n'augmente pas l'espace blanc, mais crée une cellule vide.



Cellule vide

Pour créer une cellule vide qui affiche cependant un écartement entre ses bordures, insérez dans celle-ci un saut de ligne (`<BR>`).



XHTML

Souvenez-vous : en XHTML, vous devez saisir `<br />`.

7 Enregistrez votre fichier, puis examinez-le dans un navigateur.

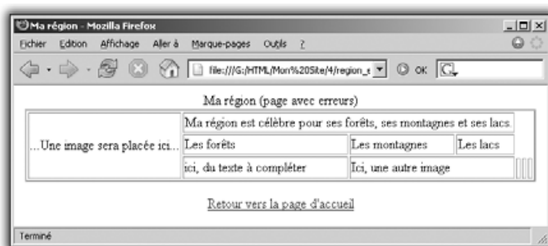


Figure 4.39 : *Aspect de la page region_erreurs après l'étape 7*

Curieux, non ? Il y a bien une erreur, mais pas celle à laquelle vous vous attendiez. Plutôt que de voir une nouvelle ligne à laquelle manque une cellule, trois cellules sont ajoutées à la droite de la dernière ligne…

Réfléchissez : si le tableau comportait théoriquement quatre lignes de quatre cellules (reportez-vous à la figure page 110), il a été précisé que la quatrième ligne était créée implicitement, dans la mesure où toutes les cellules de la ligne précédente (la troisième) s'étendaient sur deux lignes (en réalité, quatre pour la première cellule, qui débute sur la première ligne). Le Listing page 110 ne présente effectivement que trois éléments `TR`. Lorsque vous ajoutez un nouvel élément `TR`, celui-ci est considéré comme étant la quatrième ligne, possédant déjà quatre cellules définies. Les trois nouvelles cellules vont donc s'ajouter à la droite de celles-ci, sur cette même ligne, d'où l'aspect affiché.

8 Ajoutez un élément ligne vide (`<TR></TR>`) avant la balise d'ouverture de celui que vous venez d'insérer. Enregistrez votre fichier, puis examinez-le dans un navigateur.

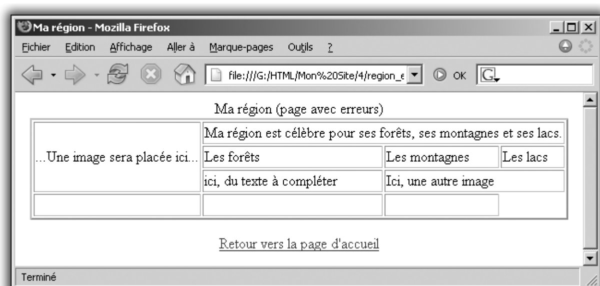


Figure 4.40 : *Aspect de la page region_erreurs après l'étape 6*

Une ligne supplémentaire apparaît bien : elle comporte une cellule de moins que ce qui est normalement nécessaire, comme vous vous y attendiez.

Remarquez que cet exemple démontre qu'il est inutile, voire dangereux, de créer des cellules s'étendant sur plusieurs lignes lorsque toutes les cellules d'une ligne s'étendent vers le bas d'un même incrément : cela reste invisible à l'écran et risque d'entraîner par la suite des erreurs, en cas de modification de la page. Mieux vaudrait ici supprimer l'attribut `rowspan="2"` dans les cellules de la troisième ligne, et modifier l'attribut de la première cellule en `rowspan="3"`. Vous pourrez alors supprimer en toute sécurité l'élément ligne vide. Faites-le donc.

8 Vous allez maintenant définir des cellules qui se chevauchent. Modifiez le second élément `TD` du second élément `TR` (donc en réalité la troisième cellule de la seconde ligne) :

```
<TD rowspan="2">Les montagnes</TD>
```

9 Enregistrez votre fichier, puis examinez-le dans votre navigateur.

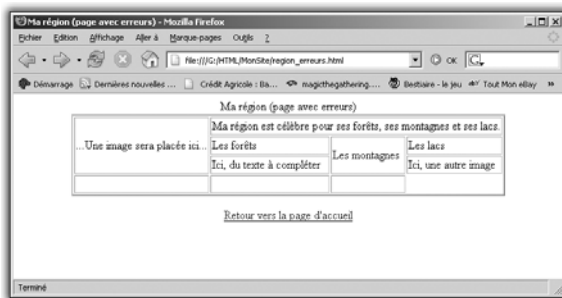


Figure 4.41 : *Aspect de la page region_erreurs après l'étape 8*

En théorie, cette cellule se recoupe désormais avec la troisième cellule de la troisième ligne. En pratique, la restitution de cette erreur (car c'en est une) peut varier selon le mode de gestion de l'agent utilisateur concerné. Ici, pratiquement quel que soit le navigateur employé, la cellule « Les montagnes » s'étend vers le bas, la cellule normalement située au-dessous étant repoussée vers la droite et semblant ne plus s'étendre latéralement sur deux cellules. Cela est d'ailleurs faux : vérifiez-le.

10 Ajoutez une cellule vide (`<TD><BR></TD>`) après la cellule « `Les lacs` » :

```
<TD>Les lacs</TD><TD><BR></TD>
```

11 Enregistrez votre fichier, puis examinez-le dans votre navigateur.

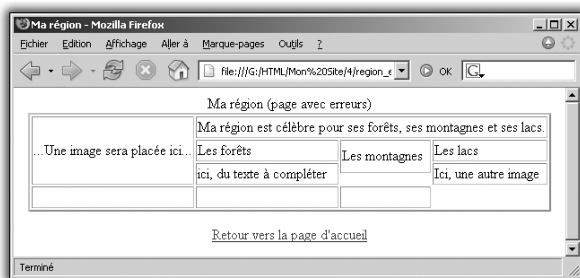


Figure 4.42 : *Aspect de la page region_erreurs après l'étape 10*

Vous voyez, grâce à l'apparition de la nouvelle cellule, que la cellule « Ici, une autre image » s'étend bien sur deux cellules. Le code de ce tableau prodigieusement laid, puisque bourré d'erreurs, est présenté dans le listing suivant, comme exemple à ne surtout pas suivre : ne créez jamais de cellules se recouvrant !

Listing 4-5 : Fichier region_erreurs.html.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"><HTML>
<HTML>
  <HEAD>
    <TITLE>Ma région (page avec erreurs)</TITLE>
    <META name="author" content="Justin Bogue">
    <META name ="version" content="1.4.1">
  </HEAD>
  <BODY>
    <TABLE border="2" align="center"
      summary="Présentation de ma région">
      <CAPTION>Ma région (page avec erreurs)</CAPTION>
      <TR>
        <TD rowspan="3">...Une image sera placée ici...</TD>
        <TD colspan="3">Ma région est célèbre pour ses forêts,
        ses montagnes et ses lacs.</TD>
      </TR>
      <TR>
        <TD>Les forêts</TD>
        <TD rowspan="2">Les montagnes</TD>
        <TD>Les lacs</TD><TD><BR></TD>
      </TR>
      <TR>
        <TD>ici, du texte à compléter </TD>
        <TD colspan="2">Ici, une autre image </TD>
      </TR>
      <TR>
        <TD><BR></TD>
        <TD><BR></TD>
```

```
        <TD><BR></TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

Regroupement de lignes : éléments THEAD, TFOOT et TBODY

Il est possible de regrouper les lignes d'un tableau dans des sections d'en-tête, de pied et de corps (à l'aide respectivement des éléments THEAD, TFOOT et TBODY). Ces regroupements apportent des informations de structure supplémentaires, qui peuvent être utilisées par les agents utilisateurs pour souligner cette structure. Les agents utilisateurs peuvent exploiter le découpage en-tête/corps/pied de façon à faire défiler le corps indépendamment des sections d'en-tête et de pied. Pour l'impression d'un grand tableau, les informations d'en-tête et de pied peuvent être répétées sur chacune des pages qui contiennent les données du tableau.

Chacun de ces éléments encadre au moins un élément TR. Ils possèdent tous les attributs facultatifs standard ainsi que les attributs d'alignement, qui seront alors appliqués à toutes les cellules du groupe de lignes.

Si vous utilisez ces éléments, l'élément TFOOT doit précéder l'élément TBODY dans votre code HTML, afin que les navigateurs Web puissent afficher les informations du pied avant de charger la totalité des lignes de données.

Les sections THEAD, TFOOT et TBODY doivent contenir le même nombre de colonnes. Plusieurs éléments TBODY peuvent coexister. L'exemple suivant illustre l'ordre et la structure des en-têtes, pieds et corps de tableau.

```
<TABLE>
<THEAD>
     <TR> ...informations d'en-tête...
</THEAD>
<TFOOT>
     <TR> ...informations de pied...
</TFOOT>
<TBODY>
     <TR> ...première rangée des données du bloc 1...
     <TR> ...seconde rangée des données du bloc 1...
</TBODY>
<TBODY>
```

```
     <TR> ...première rangée des données du bloc 2...
     <TR> ...deuxième rangée des données du bloc 2...
     <TR> ...troisième rangée des données du bloc 2...
</TBODY>
</TABLE>
```

Nous allons examiner un exemple plus concret. Supposons que dans vos passions figurent les ouvrages de science-fiction : vous avez conçu une liste d'une cinquantaine d'ouvrages, répartis en plusieurs thèmes, que vous avez présentée en tableau. Celui-ci se trouve dans une page nommée *SF.html*, qui est connectée à votre page *Mes passions*. Par pure charité, nous vous épargnerons la saisie de ce tableau, pour ne vous en présenter que le résultat.

Figure 4.43 : *Page avec tableaux renfermant des éléments THEAD, TBODY et TFOOT.*

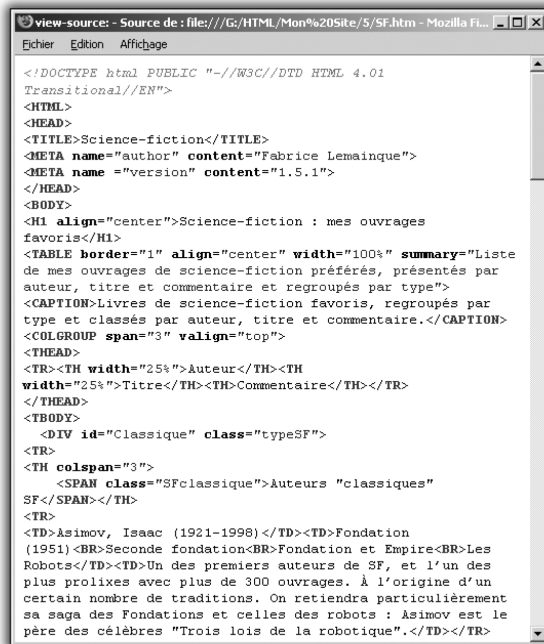
En apparence, ces regroupements sont sans effet : ils n'entrent en jeu que pour un affichage progressif et une impression. La figure suivante présente un extrait du code source de la page (voir Figure 4.44).

La balise ouvrante de l'élément `TBODY` est toujours obligatoire si le tableau possède un en-tête ou un pied. Elle peut être omise en leur absence. Les balises fermantes des trois éléments `TBODY` peuvent être omises, mais cela reste déconseillé.

Regroupement de colonnes : éléments COL et COLGROUP

Comme vous devez l'avoir désormais bien compris, HTML regroupe les cellules de données en lignes à l'aide de l'élément `TR`. Chaque élément `TR` signale une nouvelle ligne du tableau, ces lignes se présentant en

séquence, du haut vers le bas. Il n'existe pas d'élément correspondant TC pour les colonnes d'un tableau : il est par conséquent parfois délicat de placer des données en colonnes dans un tableau : il faut s'assurer que chaque ligne possède suffisamment de cellules, et que chaque colonne contient bien les bonnes données.




```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<HTML>
<HEAD>
<TITLE>Science-fiction</TITLE>
<META name="author" content="Fabrice Lemainque">
<META name ="version" content="1.5.1">
</HEAD>
<BODY>
<H1 align="center">Science-fiction : mes ouvrages
favoris</H1>
<TABLE border="1" align="center" width="100%" summary="Liste
de mes ouvrages de science-fiction préférés, présentés par
auteur, titre et commentaire et regroupés par type">
<CAPTION>Livres de science-fiction favoris, regroupés par
type et classés par auteur, titre et commentaire.</CAPTION>
<COLGROUP span="3" valign="top">
<THEAD>
<TR><TH width="25%">Auteur</TH><TH
width="25%">Titre</TH><TH>Commentaire</TH></TR>
</THEAD>
<TBODY>
  <DIV id="Classique" class="typeSF">
<TR>
<TH colspan="3">
    <SPAN class="SFclassique">Auteurs "classiques"
SF</SPAN></TH>
<TR>
<TD>Asimov, Isaac (1921-1998)</TD><TD>Fondation
(1951)<BR>Seconde fondation<BR>Fondation et Empire<BR>Les
Robots</TD><TD>Un des premiers auteurs de SF, et l'un des
plus prolixes avec plus de 300 ouvrages. À l'origine d'un
certain nombre de traditions. On retiendra particulièrement
sa saga des Fondations et celles des robots : Asimov est le
père des célèbres "Trois lois de la robotique".</TD></TR>
```

Figure 4.44 : *Code source de la page sf.html*

HTML 4.0 permet toutefois d'effectuer un regroupement structurel de colonnes à l'aide de l'élément COLGROUP et dans le but d'une mise en forme à l'aide de l'élément COL. Cette structure peut être mise en évidence grâce aux feuilles de style ou à des attributs HTML (comme l'attribut rules pour l'élément TABLE).

Un tableau contient soit un unique groupe de colonnes implicite (pas d'élément COLGROUP), soit un certain nombre de groupes de colonnes explicites, délimités chacun par une instance de l'élément COLGROUP.

Quand il est nécessaire d'isoler une ou plusieurs colonnes (par exemple, pour une information de style ou de largeur spécifique) au sein d'un groupe, les auteurs doivent identifier cette colonne avec un élément COL. L'élément COL permet de partager des attributs entre plusieurs colonnes, sans faire appel à un regroupement structurel.

Ces deux éléments possèdent un attribut capital, `span`, dont la valeur est un entier supérieur à 0 et qui spécifie le nombre de colonnes concernées. Sa valeur par défaut est `1` : c'est cette valeur qui est employée lorsque l'attribut est absent, ce qui correspond à un groupe ne contenant qu'une seule colonne.

Lorsque cet élément est présent dans un élément `COLGROUP`, les agents utilisateurs doivent l'ignorer si l'élément contient un ou plusieurs éléments `COL`.

L'intérêt de l'attribut `span` est qu'il permet de regrouper les informations concernant les largeurs de colonnes. Ainsi, si une table contient *n* colonnes, chacune d'elles ayant une largeur de 30 pixels, il est plus facile d'écrire :

```
<COLGROUP span="n" width="30">…</COLGROUP>
```

que :

```
<COLGROUP>
   <COL width="30">
   <COL width="30">
   ...jusqu'à totaliser n éléments COL...
</COLGROUP>
```

Pour appliquer une information de style particulière à la colonne *m* d'un tableau possédant *n* colonnes, vous l'isolez comme suit :

```
<COLGROUP width="20">
   <COL span="m-1">
   <COL id="mise-en-forme-particuliere">
   <COL span="n-m">
</COLGROUP>
```

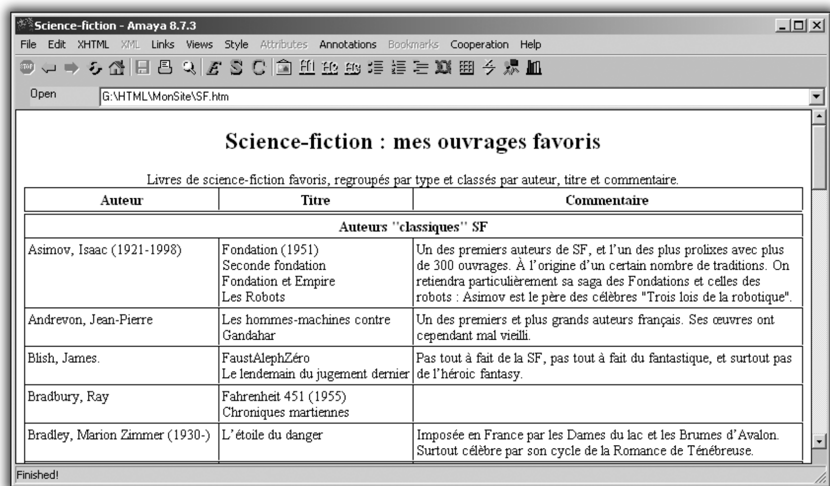
Un second attribut, `width`, spécifie la largeur par défaut de chaque colonne dans le groupe de colonnes courant. Sa valeur peut être exprimée comme habituellement en pixels, en pourcentage ou de façon relative, mais également à l'aide de la forme spéciale `"0*"` (zéro et astérisque). Celle-ci stipule que la largeur de chaque colonne du groupe doit être suffisante pour rendre le contenu de la colonne. Ceci implique que ce contenu soit connu en totalité avant que la largeur ne puisse être calculée exactement. La spécification de la valeur `"0*"` va donc empêcher l'agent utilisateur d'effectuer une restitution progressive du tableau.

Comme pour `span`, un attribut `width` présent dans un élément `COL` outrepasse celui présent dans `COLGROUP`, pour les colonnes concernées.

Il est capital de bien comprendre les rôles respectifs de ces deux éléments : si l'élément `COLGROUP` est un rassemblement structurel, l'élément `COL` rassemble les spécifications d'attributs pour les colonnes du tableau, mais ne rassemble pas les colonnes de manière structurelle. Les éléments `COL` sont vides et ne servent que de support pour les attributs. Ils peuvent apparaître à l'intérieur comme à l'extérieur d'un élément `COLGROUP`.

Ces deux éléments acceptent les attributs d'alignement horizontal et vertical déjà rencontrés (`align` et `valign`), ainsi que l'attribut `width`.

Les copies d'écran de la page 134 montrent la page précédente, *sf.html*, ayant recours à un simple élément `COL` placé après la balise ouvrante de l'élément `TABLE` (`<COL span="3" valign="top">`) pour définir l'alignement vertical dans les cellules : cela fonctionne parfaitement avec Internet Explorer et Amaya, mais malheureusement pas avec Firefox. Vous pouvez mettre en forme les colonnes en travaillant individuellement sur chaque cellule de la colonne, comme cela a été décrit précédemment : une tâche laborieuse qui augmente fortement la taille du code source, mais demeure hélas la seule solution de compatibilité avec les navigateurs non standard.



Science-fiction : mes ouvrages favoris

Livres de science-fiction favoris, regroupés par type et classés par auteur, titre et commentaire.

Auteur	Titre	Commentaire
Auteurs "classiques" SF		
Asimov, Isaac (1921-1998)	Fondation (1951) Seconde fondation Fondation et Empire Les Robots	Un des premiers auteurs de SF, et l'un des plus prolixes avec plus de 300 ouvrages. À l'origine d'un certain nombre de traditions. On retiendra particulièrement sa saga des Fondations et celles des robots : Asimov est le père des célèbres "Trois lois de la robotique".
Andrevon, Jean-Pierre	Les hommes-machines contre Gandahar	Un des premiers et plus grands auteurs français. Ses œuvres ont cependant mal vieilli.
Blish, James.	FaustAlephZéro Le lendemain du jugement dernier	Pas tout à fait de la SF, pas tout à fait du fantastique, et surtout pas de l'héroic fantasy.
Bradbury, Ray	Fahrenheit 451 (1955) Chroniques martiennes	
Bradley, Marion Zimmer (1930-)	L'étoile du danger	Imposée en France par les Dames du lac et les Brumes d'Avalon. Surtout célèbre par son cycle de la Romance de Ténébreuse.

Figure 4.45 : *Page sf.html avec recours à des éléments COL examinée sous différents navigateurs*

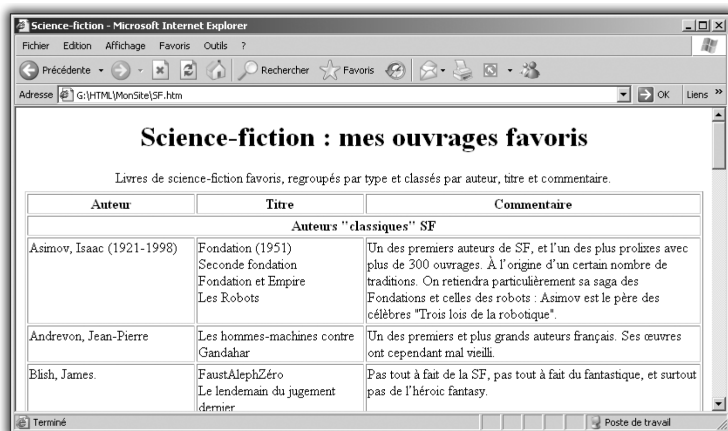


Figure 4.46 : *Sous un autre navigateur*

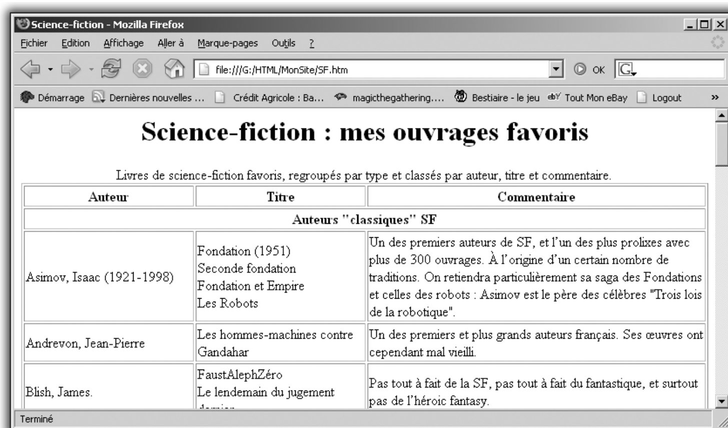


Figure 4.47 : *Sous encore un autre navigateur*



Fusion de cellules appartenant à différents regroupements

Vous devriez avoir déjà soupçonné que créer des éléments `COLGROUP` ou `COL` n'empêchait pas la fusion de cellules individuelles qui appartiennent à des éléments différents. Cela est possible, mais mieux vaut vous en abstenir. Selon l'agent utilisateur, la cellule ainsi fusionnée peut prendre les propriétés du premier élément ou du second élément de regroupement, voire empêcher simplement l'affichage correct du tableau.

Imbrication de tableaux

Il est possible d'imbriquer des tableaux : un tableau peut se trouver à l'intérieur d'une cellule d'un autre tableau. Cela peut devenir extrêmement complexe à comprendre dans le code HTML : prenez garde à bien utiliser des indentations pour savoir dans quel tableau vous vous trouvez !

Prenons un exemple simple :

1. Ouvrez le Bloc-Notes s'il ne l'est pas déjà, puis ouvrez le fichier *region.html*.
2. Comme d'habitude, modifiez le numéro de version :

```
<META name ="version" content="1.4.3">
```

3. Modifiez la légende (élément `CAPTION`) :

```
<CAPTION>Ma région (tableaux imbriqués)</CAPTION>
```

4. Comme vous l'avez fait dans la page *region_erreurs*, supprimez l'attribut `rowspan="2"` dans les cellules de la troisième ligne, et modifiez l'attribut de la première cellule en `rowspan="3"`.
5. Placez-vous dans la deuxième cellule de la troisième ligne, autrement dit dans le premier élément `TD` du dernier élément `TR`. Effacez le texte « Ici, du texte à compléter », appuyez deux fois sur ↵ et placez-vous sur la première ligne vierge.
6. Saisissez le fragment de code suivant, créant un nouveau tableau imbriqué :

```
<TABLE align="center" border="1" width="100%">
  <CAPTION>Données climatologiques</CAPTION>
<TR>
    <TH>Temp. Max.</TH><TD align="right">18 °</TD>
  </TR>
  <TR>
    <TH>Temp. min.</TH><TD align="right">-3,1 °</TD>
  </TR>
<TR>
    <TH>Pluie</TH></TH><TD align="right">682 mm</TD>
  </TR>
</TABLE>
```

7. Remarquez que nous avons retenu ici une nouvelle disposition : les en-têtes sont disposés en colonne et non plus en ligne. Enregistrez ce fichier et examinez-le dans votre navigateur.

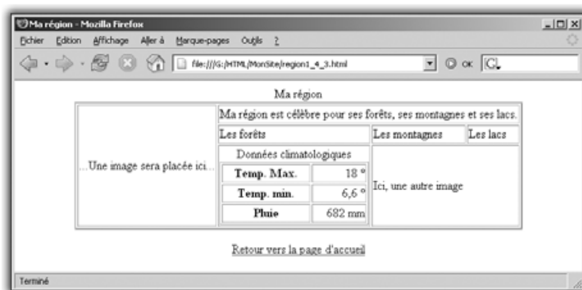


Figure 4.48 : *Tableaux imbriqués*

Autres précisions concernant les tableaux

La puissance et la richesse des tableaux font qu'ils sont largement employés comme outils de conception, et ce sur la grande majorité des sites. Cela pose encore quelques problèmes liés à la compatibilité entre navigateurs. Plus encore, en utilisant exactement la même version du même navigateur sur deux machines différentes, vous pourriez voir tout le tableau sur l'une et être obligé de faire défiler l'écran sur la seconde.

Un tableau doit être conçu avec la plus grande prudence. Par exemple, en définissant l'attribut `width` (largeur) en pixels, le tableau cesse d'être universel et n'offre un résultat parfait qu'à ceux qui possèdent exactement la même résolution d'écran et la même taille de fenêtre.

Si vous voulez absolument utiliser des tableaux pour réaliser votre mise en page, et si les feuilles de style n'apportent pas de réponse satisfaisante (ce qui serait pour le moins surprenant), veillez à définir l'attribut `width` en pourcentage, le pourcentage de la fenêtre devant être occupé par chaque colonne ou cellule.

Pour restituer un tableau, tout agent utilisateur peut effectuer les opérations suivantes :

- Restituer le contenu de l'attribut `summary`. Il est préférable de fournir une description du contenu et de la structure du tableau afin que les visiteurs qui emploient des agents utilisateurs non visuels puissent mieux la comprendre.
- Restituer la légende (élément `CAPTION`), si elle est présente.
- Restituer l'en-tête et/ou le pied du tableau (si présents). Par exemple, s'il s'agit d'un média paginé en sortie, les agents

utilisateurs peuvent placer l'en-tête en haut de chaque page et le pied en bas de chacune d'elles. De la même manière, si l'agent utilisateur fournit un mécanisme pour faire défiler les rangées, l'en-tête peut apparaître en haut de la zone qui a défilé et le pied en bas de celle-ci.

- Calculer le nombre de colonnes dans la table (c'est le nombre d'éléments `TR` de l'élément `TABLE`).
- Regrouper les colonnes en fonction des éventuelles spécifications de regroupement de colonnes.
- Restituer les cellules, rangée par rangée et regroupées dans les colonnes appropriées, entre l'en-tête et le pied. Les agents utilisateurs devraient mettre en forme le tableau d'après les attributs HTML et les spécifications de la feuille de style.

Le modèle de tableau HTML 4.01 a été conçu pour que les agents utilisateurs puissent restituer progressivement les tableaux, plutôt que de devoir attendre la réception de la totalité des données avant de commencer la restitution. Pour que cela soit possible, les concepteurs doivent indiquer le nombre de colonnes du tableau ainsi que la largeur de ses colonnes.

Plus précisément, l'agent utilisateur peut restituer un tableau en une seule passe quand les largeurs de colonnes sont spécifiées à l'aide d'une combinaison d'éléments `COLGROUP` et `COL`. Si une des colonnes est spécifiée en termes relatifs ou en pourcentage, il faut aussi spécifier la largeur du tableau en question.

Les agents utilisateurs visuels permettent aux personnes voyantes d'embrasser rapidement la structure du tableau à partir des en-têtes (`TH`) et de la légende (`CAPTION`). En revanche, ceux-ci seront souvent inadaptés pour les personnes qui dépendent d'agents utilisateurs non visuels.

Les agents utilisateurs visuels devraient éviter le rognage de toute partie du tableau contenant la légende, à moins de fournir un moyen pour accéder à toutes les parties du tableau, par exemple par défilement horizontal ou vertical.

Calcul du nombre de colonnes dans un tableau

Il existe deux façons de déterminer le nombre de colonnes dans une table (par ordre de priorité) :

1 Si l'élément `TABLE` contient des éléments `COLGROUP` ou `COL`, les agents utilisateurs calculent comme suit le nombre de colonnes :

- Pour chaque élément `COL`, prendre la valeur de son attribut `span` (la valeur par défaut est `"1"`).
- Pour chaque élément `COLGROUP` contenant au moins un élément `COL`, ignorer l'attribut `span` de l'élément `COLGROUP`. Pour chaque élément `COL`, effectuer les calculs de l'étape 1.
- Pour chaque élément `COLGROUP` vide, prendre la valeur de son attribut `span` (la valeur par défaut étant `"1"`).

2 Sinon, quand l'élément `TABLE` ne contient aucun élément `COLGROUP` ou `COL`, les agents utilisateurs devraient fonder le nombre de colonnes sur ce qui est requis par les lignes. Le nombre de colonnes est égal au nombre de colonnes requises par la ligne possédant le plus de colonnes, y compris les cellules qui couvrent plusieurs colonnes. Les lignes possédant un nombre de colonnes moindre seront garnies de cellules vides en fin de ligne. La « fin » d'une ligne dépend de la directionnalité du tableau.

Lorsqu'un tableau contient des éléments `COLGROUP` ou `COL` et que les deux types de calcul n'aboutissent pas au même nombre de colonnes, cela entraîne une erreur.

Une fois que l'agent utilisateur a calculé le nombre de colonnes dans le tableau, il peut les regrouper en groupes de colonnes.

Héritage des attributs

Le problème de l'héritage des attributs d'alignement dans un tableau peut parfois sembler complexe. Certains d'entre eux peuvent être hérités de l'élément `TABLE` ou de la ligne à laquelle la cellule appartient. Certains peuvent également être hérités d'une colonne, à l'aide d'éléments `COLGROUP` ou `COL`.

Les regroupements de colonnes apportent des informations structurelles supplémentaires qui peuvent être exploitées par les agents utilisateurs. Il est également possible de déclarer des propriétés de colonnes au début de la définition du tableau (*via* les éléments `COLGROUP` et `COL`), pour que les agents utilisateurs puissent restituer progressivement la table au lieu de devoir attendre que toutes les données du tableau soient arrivées avant de pouvoir les restituer.

L'alignement du contenu des cellules peut être spécifié cellule par cellule ou bien peut être hérité des éléments qui l'englobent tels que la rangée, la colonne ou la table même.

L'ordre de priorité (de la plus élevée à la plus basse) des attributs d'alignement (`align`, `char` et `charoff`) est le suivant :

1. L'attribut d'alignement défini sur un élément à l'intérieur des données de la cellule (par exemple, un élément `P`).
2. L'attribut d'alignement défini sur une cellule (les éléments `TH` et `TD`).
3. L'attribut d'alignement défini sur un élément de regroupement de colonnes (les éléments `COL` et `COLGROUP`). Quand la cellule appartient à un regroupement de colonnes, la propriété d'alignement est héritée de la définition de la cellule initiant le recouvrement.
4. L'attribut d'alignement défini sur un élément de ligne ou de regroupement de lignes (les éléments `TR`, `THEAD`, `TFOOT` et `TBODY`). Quand la cellule appartient à un ensemble de plusieurs lignes, la propriété d'alignement est héritée de la définition de la cellule initiant le recouvrement.
5. L'attribut d'alignement défini sur la table (l'élément `TABLE`).
6. La valeur d'alignement par défaut.

L'ordre de priorité (de la plus élevée à la plus basse) de l'attribut `valign` (ainsi que des autres attributs hérités `lang`, `dir` et `style`) est le suivant :

1. L'attribut défini sur un élément à l'intérieur des données de la cellule (par exemple, un élément `P`).
2. L'attribut défini sur une cellule (les éléments `TH` et `TD`).
3. L'attribut défini sur un élément de rangée ou de regroupement de rangées (les éléments `TR`, `THEAD`, `TFOOT` et `TBODY`). Quand la cellule fait partie d'un ensemble recouvrant plusieurs rangées, la valeur de l'attribut est héritée de la définition de la cellule initiant le recouvrement.
4. L'attribut défini sur un élément de regroupement de colonnes (les éléments `COL` et `COLGROUP`). Quand la cellule appartient à un regroupement de colonnes, la valeur de l'attribut est héritée de la définition de la cellule initiant le recouvrement.
5. L'attribut défini sur la table (l'élément `TABLE`).

6 La valeur par défaut de l'attribut.

En outre, lors de la restitution des cellules, l'alignement horizontal est déterminé par les colonnes, de préférence aux lignes, tandis que les lignes ont la préférence sur les colonnes pour l'alignement vertical.

Il existe un attribut non encore examiné, l'attribut `dir`. Celui-ci définit la *directionnalité* d'un tableau (qui, par défaut, est de gauche à droite). Pour une table de gauche-à-droite, la colonne zéro se trouve sur le côté gauche et la rangée zéro se trouve en haut. Pour une table de droite-à-gauche, la colonne zéro se trouve sur le côté droit et la rangée zéro en haut. Lorsqu'il est spécifié pour l'élément `TABLE`, l'attribut `dir` affecte aussi la direction du texte à l'intérieur des cellules du tableau (puisqu'il est hérité par les éléments de type bloc).

Pour spécifier une table de droite-à-gauche, fixez la valeur de l'attribut `dir` comme suit :

```
<TABLE dir="RTL">
...le reste du tableau...
</TABLE>
```

Il est possible de modifier la direction du texte dans une cellule individuelle en spécifiant l'attribut `dir` dans l'élément qui définit cette cellule.



Élément TABLE

L'élément `TABLE` est le seul pour lequel l'attribut `dir` inverse l'ordre visuel des colonnes : une ligne de tableau (`TR`) ou un groupe de colonnes (`COLGROUP`) seuls ne peuvent être inversés indépendamment.

Remarquez bien qu'une cellule peut hériter de la valeur d'un attribut non de son parent, mais de la première cellule d'un recouvrement. Il s'agit d'une exception aux règles générales d'héritage des attributs.

Autres mises en forme de tableaux : attributs char et charoff

Lors de la première publication de la spécification, en 1997, la spécification des feuilles de style CSS1 n'offrait pas de mécanismes permettant le contrôle de tous les aspects de la mise en forme visuelle des tables. Depuis lors, la spécification des feuilles de style CSS2 a ajouté des propriétés qui permettent la mise en forme visuelle des tables.

L'attribut d'alignement `align` employé dans un tableau peut posséder une valeur particulière (`align="char"`), qui stipule que l'alignement est effectué sur un caractère unique présent dans un fragment de texte qui sert d'axe d'alignement.

Ce caractère est défini par l'attribut `char`, dont la valeur par défaut est le séparateur point décimal pour la langue courante, telle qu'elle est définie par l'attribut `lang` (par exemple, le point « . » pour l'anglais et la virgule « , » pour le français).

L'attribut `charoff`, quand il est présent, spécifie le décalage de la première occurrence du caractère d'alignement sur chaque ligne. Si la ligne ne comprend pas de caractère d'alignement, elle devrait glisser horizontalement de manière à finir sur la position d'alignement.

Lors du recours à l'attribut `charoff` pour fixer le décalage du caractère d'alignement, la direction du décalage est déterminée par la direction de texte courante (telle que spécifiée par l'attribut `dir`). Dans les textes de gauche-à-droite (la valeur par défaut), le décalage s'opère à partir de la marge de gauche. Inversement, pour les textes de droite-à-gauche, le décalage s'effectue à partir de la marge de droite.

Hélas ! Si la spécification mentionne effectivement que les agents utilisateurs ne sont pas obligés de gérer ces attributs, aucun des navigateurs testés n'a été en mesure de le faire dans leurs versions actuelles (Firefox 2.0.0.12, Mozilla 9, Internet Explorer 7.0.5730 et Amaya 7.8.3)…

4.3. Résumé

- Il est possible de créer en HTML trois types de listes : ordonnées ou numérotées, non ordonnées ou listes à puces et listes de définitions.
- Une liste numérotée est créée à l'aide de l'élément `OL` renfermant des items de liste `LI`. Il est possible de modifier le type de numérotation à l'aide de l'attribut `type`, la valeur de début à l'aide de l'attribut `start` et de réinitialiser la numérotation dans un item de liste à l'aide de l'attribut `value`.
- Une liste à puces est créée à l'aide de l'élément `UL` renfermant des items de liste `LI`. Il est possible de modifier l'aspect de la puce à l'aide de l'attribut `type`, pour toute la liste ou pour un item particulier.

- Une liste de définitions est créée à l'aide de l'élément `DL` renfermant des éléments enfants `DT` (terme de définition) et `DD` (description de la définition).
- Il est possible d'imbriquer un nombre quelconque de listes de tous types.
- Un tableau est créé à l'aide de l'élément `TABLE`. Il contient des lignes (élément `TR`) et des colonnes créant des cellules pouvant renfermer des informations de rubrique (élément `TH`) ou des informations de données (élément `TD`).
- L'élément `TABLE` possède trois attributs, `border`, `frame` et `rules`, qui indiquent à l'agent utilisateur les bordures et les règles à restituer.
- L'élément `CAPTION` permet de doter le tableau d'une légende. Il est également utile de renseigner l'attribut `summary` de l'élément `TABLE`, au bénéfice des agents utilisateurs non visuels.
- Les attributs `cellspacing` et `cellpadding` permettent de modifier respectivement l'espacement entre les cellules, ainsi que le décalage entre le contenu de la cellule et la bordure.
- L'alignement horizontal d'un tableau s'effectue à l'aide de l'attribut `align` et l'alignement vertical grâce à l'attribut `valign`. La largeur est contrôlée par l'attribut `width`. Elle peut être exprimée de façon fixe, en pourcentage ou proportionnellement.
- Il est possible de fusionner des cellules d'une même ligne à l'aide de l'attribut `colspan` et d'une même colonne à l'aide de l'attribut `rowspan`. Ces deux attributs peuvent être combinés.
- Vous regroupez structurellement des lignes à l'aide d'éléments d'en-tête de tableau (`THEAD`), de corps de tableau (`TBODY`) et de pied de tableau (`TFOOT`).
- Vous regroupez structurellement des colonnes à l'aide d'éléments `COLGROUP` et définissez des attributs partagés à l'aide d'éléments `COL`. L'attribut `span` spécifie l'étendue d'un regroupement.
- Il est possible d'imbriquer des tableaux.
- C'est avec les tableaux qu'il est possible de constater les différences de restitution les plus importantes entre navigateurs.

Chapitre 5

Mise en forme avancée

Principaux éléments structuraux 136
Éléments de structuration du texte 149
Caractères spéciaux et encodage de caractères 156
Modification de l’apparence du texte 164
Notation mathématique 169
Résumé 173

HTML propose de nombreuses autres possibilités de mise en forme des pages ou de portions de texte, à l'aide d'éléments de type bloc ou de type ligne. Tous répondent à des besoins précis, et peuvent être employés de diverses façons, par exemple soit à la place de l'élément de paragraphe normal P, soit en combinaison avec celui-ci.

Éléments étudiés dans ce chapitre :

```
HR
<!--   -->
DIV, SPAN
PRE, TT
BLOCQUOTE, Q
CITE
ABBR, ACRONYM
EM, STRONG
OBJECT
CODE, SMP, KBD
```

5.1. Principaux éléments structuraux

Élément HR

Cet élément bloc est probablement le plus curieux de cette série, car il s'agit d'un élément vide dépourvu de balise de fermeture, dont le seul rôle est d'insérer une ligne horizontale à l'écran. Placez-le après les titres principaux et les changements de section. Une telle barre horizontale constitue une preuve manifeste de changement de sujet.

1 Ouvrez le Bloc-Notes, puis le fichier *pageacc1_4_1.html*.

2 Comme d'habitude, modifiez le numéro de version :

```
<META name="version" content="1.5.1">
```

3 Ajoutez un élément HR de barre horizontale entre les titres de niveau 1 et de niveau 2 :

```
<H1 align="center">Ma page d'accueil</H1>
<HR>
<H2 align="center">Bienvenue sur mon site.</H2>
```

4 Ajoutez également un élément HR juste avant la balise de fermeture de l'élément BODY.

```
<HR>
</BODY>
```

5 Enregistrez votre fichier sous le nom *pageacc1_5_1.html*, puis examinez-le dans votre navigateur.

La page affiche désormais deux barres horizontales.

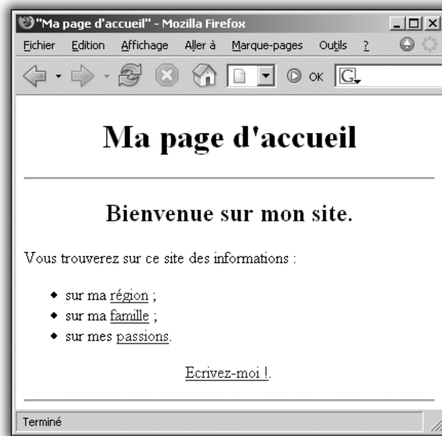


Figure 5.1 : *Barres horizontales dans la page d'accueil*



XHTML

En XHTML, comme pour tout élément vide, vous devez l'écrire `<hr />`.

Commentaire HTML

Autre bloc particulier, le cas où vous souhaitez insérer un commentaire non lu par le navigateur, et donc non affiché dans la fenêtre de celui-ci.

```
<!-- commentaire -->
```

où `commentaire` est le texte du commentaire. Ce texte peut occuper plusieurs lignes. Les commentaires sont fréquemment saisis en majuscules afin de faciliter leur identification dans le code.

Prenons un exemple concret : vous possédez un en-tête ou un pied de page commun à toutes vos pages Web et savez que vous allez devoir régulièrement le modifier. Vous pouvez en faciliter grandement la recherche en insérant juste avant un commentaire comme :

```
<!-- DEBUT DU PIED DE PAGE -->
```

Les commentaires sont précieux lorsqu'un site Web est développé en collaboration, pour expliquer certains ensembles de code ou simplement pour servir d'aide-mémoire ou de lignes directrices en vue d'améliorations ultérieures.

1 Revenez au fichier *pageacc1_5_1.html* actuellement ouvert dans le Bloc-Notes.

2 Modifiez à nouveau le numéro de version :

```
<META name="version" content="1.5.2">
```

3 Ajoutez un élément de commentaire juste après le titre de niveau 2 :

```
<H2 align="center">Bienvenue sur mon site.</H2>
<!-- Contenu à étoffer, probablement à l'aide d'une
barre de navigation.
Remarque importante : ne pas oublier de modifier les
liens tant de la page d'accueil que des pages
secondaires en cas de modification des noms de
fichier ! -->
```

4 Ajoutez un autre commentaire juste avant la balise de fermeture de l'élément `BODY`.

```
<HR>
<!-- Des remarques légales ne seraient pas
superflues -->
</BODY>
```

Remarquez que le premier commentaire s'étend sur plusieurs lignes, contrairement au second.

5 Enregistrez votre fichier sous le nom *pageacc1_5_2.html*, puis examinez-le dans votre navigateur : l'aspect est strictement identique à la page précédente, puisque les commentaires sont ignorés par l'agent utilisateur.

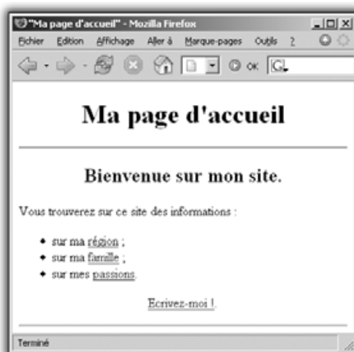


Figure 5.2 :
Les commentaires n'apparaissent pas dans la fenêtre du navigateur



Couples de traits d'union

Les espaces vierges sont interdits entre le délimiteur d'ouverture de la déclaration de balisage <! et le délimiteur d'ouverture de commentaire ––, mais autorisés entre le délimiteur de fin de commentaire –– et le délimiteur de fin de déclaration de balisage >. Une erreur classique consiste à inclure une chaîne de traits d'union à l'intérieur d'un commentaire. Prenez garde à ne pas insérer une suite de deux traits d'unions adjacents ou plus à l'intérieur d'un commentaire : l'agent utilisateur l'interpréterait comme la fin du commentaire !

Les commentaires se révèlent également précieux justement en raison de leur capacité à ne pas être interprétés par l'agent utilisateur : lorsque vous testez le code d'une nouvelle page et rencontrez une erreur, il est parfois utile de *mettre en commentaire* une portion du code, qui sera ainsi ignorée par le navigateur. Il peut s'agir d'une portion dont vous savez qu'elle fonctionne correctement ou au contraire d'une partie de posant problème. Prenons un exemple.

1. Revenez au fichier *pageacc1_5_2.html* actuellement ouvert dans le Bloc-Notes.
2. Modifiez à nouveau le numéro de version :

```
<META name="version" content="1.5.3">
```

3. Déclarez comme commentaire le paragraphe d'envoi de courriel :

```
<!--
<P align="center">
<A href="mailto:votre_nom@votre_FAI">Ecrivez-moi !</A>
</P>
-->
```

4. Enregistrez votre fichier sous le nom *pageacc1_5_3.html*, puis examinez-le dans votre navigateur : l'ancre permettant l'envoi d'un courriel à votre adresse n'apparaît plus dans la fenêtre (voir Figure 5.3).

Cette technique est extrêmement intéressante : pensez par exemple à un tableau s'affichant d'une façon qui ne correspond pas à votre attente. Placez en commentaire tous les éléments TR, puis sortez-les un à un du commentaire en déplaçant la balise <!–– jusqu'à identifier celui qui pose problème !

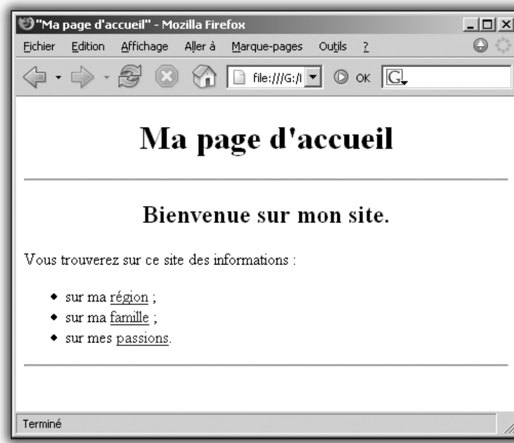


Figure 5.3 : *La ligne d'envoi de courriel n'apparaît plus sur la page : le code placé en commentaire est ignoré.*

Éléments DIV et SPAN

Les éléments `DIV` et `SPAN`, employés avec leurs attributs `id` et `class`, offrent un mécanisme générique qui rajoute de la structure aux documents. L'élément `DIV` est un élément bloc, tandis que `SPAN` est un élément ligne. Impliqués uniquement dans un but structurel, ils n'imposent en eux-mêmes aucune présentation particulière du contenu. Ils sont essentiellement employés avec les feuilles de style et/ou l'attribut `lang` pour mieux exploiter HTML.

Prenons un exemple concret. Souvenez-vous que, lors de l'étude des en-têtes, corps et pieds de tableaux (`THEAD`, `TBODY` et `TFOOT`), nous avions supposé que l'une de vos passions concernait la littérature de science-fiction. Vous aviez (théoriquement) conçu un tableau présentant différents ouvrages. HTML ne disposant pas d'élément identifiant des objets comme « SF classique », « Heroic fantasy », etc., les éléments `DIV` et `SPAN` permettent d'obtenir les effets de structure et de présentation souhaités. Voici comment vous pourriez vous en servir :

```
<TBODY>
  <DIV id="Classique" class="typeSF">
  <TR>
    <TH colspan="3">
    <SPAN class="SFclassique">Auteurs "classiques" SF</SPAN>
    </TH>
  </TR>
...autre contenu...
   </DIV>
```

```
</TBODY>
<TBODY>
  <DIV id="Heroic" class="typeSF">
  <TR>
    <TH colspan="3">
    <SPAN class="heroic">Auteurs "heroic-fantasy"</SPAN>
    </TH>
  </TR>
...autre contenu...
   </DIV>
</TBODY>
...
```

Vous avez défini des blocs à présentation particulière, liés au style attribué à la classe *typeSF*, et des éléments ligne également à présentation particulière, liés aux classes *Sfclassique* et *heroic*.

Observée dans un navigateur, cette page ne présente aucune différence avec la version précédente (reportez-vous à la figure de la page 134), puisque vous n'avez pas encore défini ces classes de feuilles de style. Vous verrez plus en détail dans le Chapitre 8, traitant des feuilles de style, comment exploiter ces structures.

Élément PRE

L'élément `PRE` est très intéressant en ce qu'il définit un bloc de texte comme préformaté. Les agents utilisateurs visuels (les navigateurs) gèrent comme suit le contenu d'un élément `PRE` :

- Ils peuvent respecter les espaces vierges présents dans le texte.
- Ils peuvent afficher le texte à l'aide d'une police à espacement non proportionnel.
- Ils peuvent désactiver le retour à la ligne automatique.
- Ils ne doivent pas désactiver le traitement bidirectionnel.

Remarquez bien que les navigateurs ne sont donc pas tenus de respecter absolument les espaces vierges présents dans le contenu d'un élément `PRE`.

Pour examiner un exemple, supposons que parmi vos passions figure également Shakespeare et, plus particulièrement parmi les œuvres de cet éminent auteur, Hamlet. Une de vos pages contient un extrait de cette pièce : vous souhaitez respecter exactement la mise en forme de l'auteur.

1 Plutôt que de créer dans le Bloc-Notes un nouveau fichier, repartez du fichier précédent (en principe *pageacc1_5_2.html*) en supprimant tout le contenu de l'élément `BODY`. Cela évite d'avoir à saisir de nouveau tous les éléments communs.

2 Dans l'élément `HEAD`, modifiez l'élément `TITLE` :

```
<TITLE>"Hamlet de Shakespeare"</TITLE>
```

3 Puis modifiez le numéro de version :

```
<META name="version" content="1.5.1">
```

4 Ajoutez un nouveau titre de niveau 1 dans l'élément `BODY` :

```
<H1 align="center">Extrait de Hamlet de Shakespeare</H1>
```

5 Ajouter ensuite une barre horizontale, puis un paragraphe :

```
<HR>
<P>Ceci est un de mes extraits favoris : un petit air un
 peu coquin chanté par Ophélia à ses parents</P>.
```

6 Créez ensuite un élément `PRE` et placez-y son contenu, puis faites-le suivre d'une autre barre horizontale :

```
<PRE>
Alors, il se leva et mit ses habits,
          Et ouvrit la porte de sa chambre ;
Et vierge elle y entra, et puis oncques vierge
          Elle n'en sortit.
</PRE>
<HR>
```

7 Enregistrez votre fichier sous le nom *hamlet1_5_1.html*, puis examinez-le dans votre navigateur : l'aspect du texte est strictement respecté par l'agent utilisateur.



Figure 5.4 :
Aspect d'un élément PRE

L'élément PRE possède un élément ligne analogue, TT.

Éléments BLOCKQUOTE et Q

Nous venons de placer sur une page Web un extrait de texte célèbre : il est normalement légalement obligatoire de citer les sources de cette œuvre. Il est fréquent de devoir placer sur une page Web des citations. Une citation peut émaner d'un avis reconnu pour souligner l'intérêt d'un produit, peut indiquer une source d'information ou représenter des notes de lecture. Elle peut aussi simplement être une de vos citations favorites, affichée quelque part sur votre page Web.

Quel que soit le motif de sa présence, une citation se doit de respecter un style bien défini pour apparaître de façon satisfaisante sur tous les navigateurs. Tel est le but de l'élément BLOCKQUOTE, un élément de type bloc, dont l'équivalent en type ligne est l'élément Q. Les agents utilisateurs visuels restituent en général l'élément BLOCKQUOTE sous la forme d'un bloc de texte en retrait. Voici la syntaxe des éléments BLOCKQUOTE et Q :

```
<BLOCKQUOTE [lang="code_langue"][cite="URI_source"]>
<Q [lang="code_langue"][cite="URI_source"]>
```

1 Revenez au fichier précédent, et modifiez le numéro de version :

```
<META name="version" content="1.5.2">
```

2 Encadrez l'élément PRE par un élément BLOCKQUOTE :

```
<BLOCKQUOTE>
<PRE>
... texte ...
</PRE>
</BLOCKQUOTE>
```

3 Enregistrez votre fichier sous le nom *hamlet1_5_2.html*, puis examinez-le dans votre navigateur. La différence est faible par rapport à la figure précédente, mais le léger retrait attendu est bien présent (voir Figure 5.5).

L'attribut facultatif cite de l'élément BLOCKQUOTE permet de spécifier l'URI du document source (si celui-ci est disponible sur le Web). Mieux vaut toutefois employer l'élément CITE pour indiquer la source de la citation. Cet élément apparaît sur une ligne individuelle, le plus souvent en italique.

4 Revenez à votre fichier, puis ajoutez sous l'élément BLOCKQUOTE l'élément suivant :

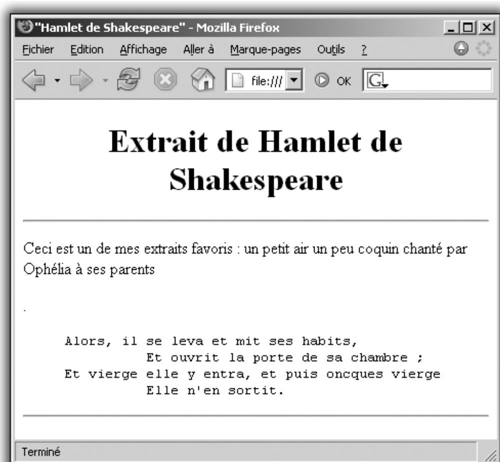


Figure 5.5 : *Aspect d'un élément BLOCKQUOTE combiné avec un élément PRE*

```
<CITE>Hamlet, Acte IV Scène V.William Shakespeare,
"Richard III, Roméo et Juliette, Hamlet", Trad.
F-V Hugo.1979, Ed. Garnier-Flammarion,
ISBN 2-08-070006-5, page 339.</CITE>
```

5 Enregistrez votre fichier sans modifier son nom, puis examinez-le dans votre navigateur. La source de la citation est bien indiquée sous la forme attendue.

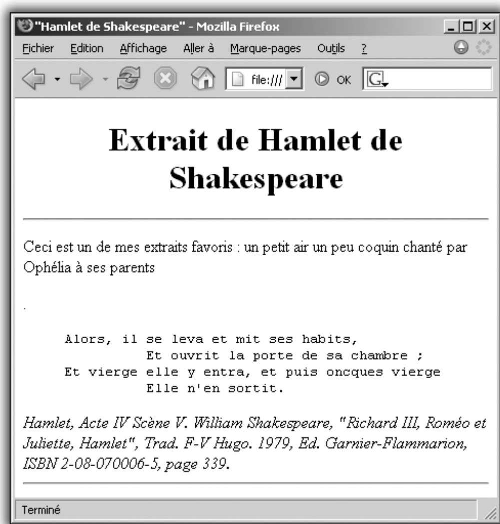


Figure 5.6 : *Aspect d'un élément BLOCKQUOTE combiné avec un élément PRE*

Remarquez que vous pourriez améliorer la mise en forme, par exemple en plaçant des sauts de ligne dans la source et/ou en employant l'attribut `align`. Comme vous savez désormais le faire, nous vous en laissons l'initiative comme exercice complémentaire. Le code complet de cet exemple est présenté dans le listing suivant.

Listing 5-1 : Code du fichier Hamlet 1.5.2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>"Hamlet de Shakespeare"</TITLE>
    <META name="author" content="Fabrice Lemainque">
    <META name="version" content="1.5.2">
  </HEAD>
  <BODY>
    <H1 align="center">Extrait de Hamlet de Shakespeare</H1>
    <HR>
    <P>Ceci est un de mes extraits favoris : un petit air un
peu coquin chanté par Ophélia à ses parents</P>.
    <BLOCKQUOTE>
      <PRE>
Alors, il se leva et mit ses habits,
          Et ouvrit la porte de sa chambre ;
Et vierge elle y entra, et puis oncques vierge
          Elle n'en sortit.
      </PRE>
    </BLOCKQUOTE>
    <CITE>Hamlet, Acte IV Scène V. William Shakespeare,
"Richard III, Roméo et Juliette, Hamlet", Trad. F-V Hugo.
1979, Ed. Garnier-Flammarion, ISBN 2-08-070006-5, page
339.</CITE>
    <HR>
  </BODY>
</HTML>
```

Le rôle de l'attribut `lang`, également facultatif, est plus subtil. Des citations peuvent être imbriquées, ce qui signifie que vous pouvez utiliser l'élément `Q` à l'intérieur d'un élément `BLOCKQUOTE`. Un élément `Q` peut d'ailleurs également être placé à l'intérieur d'un autre élément `Q`.

La spécification est toutefois un peu contradictoire en ce qui concerne le comportement des navigateurs vis-à-vis du contenu d'un élément `Q` : en principe, celui-ci devrait être restitué avec des marques de citation englobantes, si bien que les auteurs ne devraient donc pas placer de marque de citation au début ni à la fin du contenu d'un élément `Q`. Les marques de citation employées doivent dépendre de la langue, telle que spécifiée par l'attribut `lang` : en effet, le type de marque de citation,

extérieure ou intérieure (imbriquées), dépend de la langue. Ceci devrait être respecté par les agents utilisateurs.

Il est recommandé que les mises en œuvre de feuilles de style fournissent un moyen d'ajouter des marques de citation avant et après une citation encadrée par l'élément BLOCKQUOTE, d'une manière appropriée au contexte de langue courant et au degré d'imbrication des citations. Mais certains auteurs employant à tort l'élément BLOCKQUOTE pour obtenir un texte en retrait, les agents utilisateurs ne devraient finalement pas ajouter de marques de citation en style par défaut, afin de préserver les intentions de ces auteurs.

Tout ceci n'est pas très clair. Mieux vaut considérer un exemple. Les Anglo-Saxons se servent de guillemets comme marques de citation extérieures et d'apostrophes comme marques de citation intérieures, tandis que les francophones privilégient des guillemets imbriqués. De ce fait, le fragment de code suivant :

Listing 5-2 : Citations imbriquées

```
<Q lang="en-us">He shouted
<Q lang="en-us">What are you doing here?</Q></Q>
<Q lang="fr">Il s'écria
<Q lang="fr">Que faites-vous ici ?</Q></Q>
```

devrait être affiché comme ceci :

```
"He shouted 'What are you doing here?'''
"Il s'écria "Que faites-vous ici ?""
```

Mais comme le montrent les trois copies d'écran des figures suivantes, le rendu des navigateurs, même récents, manque pour le moins de cohérence :

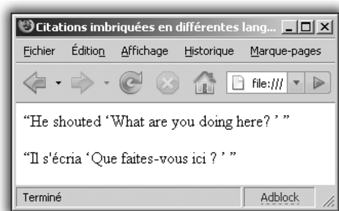


Figure 5.7 :
Rendu de citations imbriquées dans le navigateur Firefox 2.0.

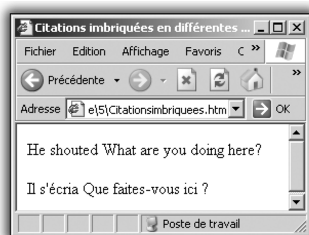


Figure 5.8 :
Rendu de citations imbriquées dans le navigateur Internet Explorer (6 et 7).

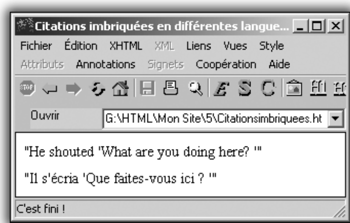


Figure 5.9 :
Rendu de citations imbriquées dans le navigateur Amaya.

Mieux vaut donc observer la plus grande prudence en cas d'imbrication de citations. La sagesse recommande d'employer de préférence les entités caractères HTML étudiées dans la suite de ce chapitre pour afficher guillemets ou apostrophes, plutôt que d'imbriquer des éléments `Q`.

Par ailleurs, l'emploi des éléments `BLOCKQUOTE` et `Q` uniquement pour obtenir le retrait d'un texte est déconseillé : préférez le recours aux feuilles de style.

Élément ADDRESS

L'élément `ADDRESS`, généralement placé au début ou à la fin d'un document, sert à fournir des informations de contact applicables au document ou à une partie essentielle de celui-ci. Ses balises d'ouverture et de fermeture sont toutes deux obligatoires.

Un exemple concret semble préférable à une longue explication.

1 Ouvrez dans le Bloc-Notes le fichier *pageacc1_5_2.html* (c'est en principe actuellement *hamlet1_5_2.html* qui est ouvert).

2 Modifiez à nouveau le numéro de version :

```
<META name="version" content="1.5.4">
```

3 Entrez le dernier élément `HR` et le dernier commentaire, et insérez un élément `ADDRESS` comme suit :

```
<ADDRESS>
<A href="../ORignal/">Olivier Rignal</A> et
<A href="../JBogue/">Justin Bogue</A> sont les
personnes à contacter pour tout <A href="probleme">
problème</A> rencontré sur le site.
</ADDRESS>
```

4 Enregistrez votre fichier sous le nom *pageacc1_5_4.html*, puis examinez-le dans votre navigateur : la page propose désormais des informations de contact.

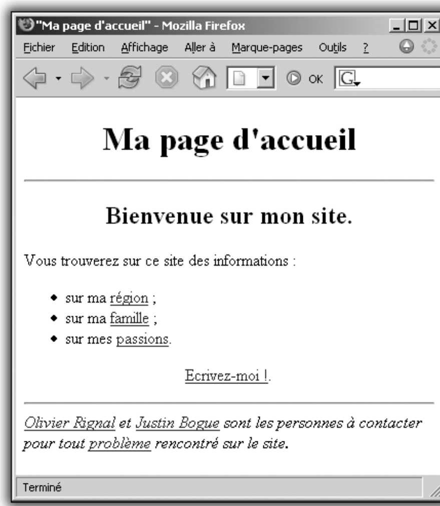


Figure 5.10 : *Élément ADDRESS*

Remarquez que, dans cet exemple, les valeurs des attributs `href` sont toutes inexistantes : il convient de créer les destinations de lien adéquates.

Modification des marges d'un paragraphe

Il est parfois souhaitable ou nécessaire de modifier les marges d'un bloc de texte (élément `DIV`, `P` ou `PRE`). Vous pouvez y parvenir à l'aide de l'attribut `style` placé à l'intérieur de la balise d'ouverture de l'élément concerné :

```
<P style="margin-left: 10pt">
```

Cela crée une marge supplémentaire gauche de 10 points. Remarquez que la syntaxe de l'attribut `style` est la même que celle de n'importe quel attribut HTML : `attribut="valeur"`. Il est applicable à la

plupart des éléments HTML (reportez-vous à l'Annexe A), mais est désormais déconseillé au profit des feuilles de style.



Taille en pixels

Pour calculer avec plus de précision l'importance d'une taille exprimée en pixels, utilisez la formule suivante :

Taille réelle = valeur × (*largeur de l'écran*)/(*résolution horizontale de l'écran*)

En supposant que vous possédiez un écran de 17' mesurant (environ) 34 cm de large et employé avec une résolution de 1280 × 1024 (donc 1280 pixels de résolution horizontale), une valeur de 10 pixels correspond à :

10 × 34/1280 = 0,266 centimètres soit 2,7 mm

La chose est plus délicate pour les mesures exprimées en points : si sur un Mac, la règle est « un point = un pixel », sur un PC, le réglage par défaut est « un point = 4/3 pixel ».

C'est la raison pour laquelle certains auteurs écrivent que le Mac a une dpi de 72 et le PC une dpi de 96. Tout dépend de la définition du « dot » de dpi (pixel ou point).

Puisque nous parlons essentiellement de PC, une valeur de 10 points correspond à environ 3,5 mm.

5.2. Éléments de structuration du texte

HTML propose un certain nombre d'éléments de type ligne permettant de caractériser précisément une partie de texte. Ces éléments sont présentés dans le tableau de la page 155.

Gardez bien présent à l'esprit que le but de ces éléments est avant tout de contribuer à la *structure* du texte, non d'obtenir une présentation particulière.

Éléments EM et STRONG

Les éléments `EM` (*EMphasis*, emphase) et `STRONG` (fort) s'utilisent pour mettre en exergue une portion du texte qui possède, par exemple, une signification particulière dans un contexte donné.

La présentation des éléments de phrase dépend de l'agent utilisateur. En général, les agents utilisateurs visuels présentent le texte de l'élément `EM`

en italique et celui de l'élément `STRONG` en gras. Les agents utilisateurs vocaux, eux, devraient changer les paramètres de la synthèse vocale tels que le volume, la hauteur ou le timbre.

1. Revenez au le fichier *pageacc1_5_4.html*, en principe celui actuellement ouvert dans le Bloc-Notes.
2. Modifiez à nouveau le numéro de version :

```
<META name="version" content="1.5.5">
```

3. Modifiez comme suit le titre de niveau 2 :

```
<H2 align="center">Bienvenue sur <EM>mon</EM> site.</H2>
```

4. Modifiez comme suit le contenu du premier élément `P` :

```
<P>Vous trouverez sur ce site des <STRONG>informations
</STRONG> :</P>
```

5. Enregistrez votre fichier sous le nom *pageacc1_5_5.html*, puis examinez-le dans votre navigateur : le mot « mon » apparaît en italique, le mot « informations » en gras.

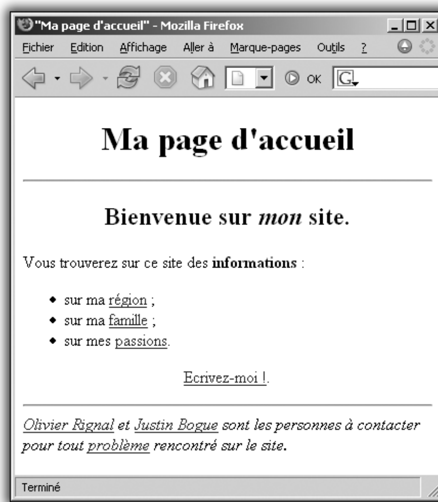


Figure 5.11 :
Éléments EM et STRONG

Vous avez pu rencontrer d'autres éléments qui permettent de placer une partie de texte en italique (élément `I`) ou en gras (élément `B`). Ces éléments, sur lesquels nous reviendrons, modifient l'aspect de la police, mais ne possèdent aucun rôle structurel. Autrement dit, un mot placé en gras ou en italique par `G` ou `I` attire visuellement l'attention, ce qui lui confère une importance particulière, alors qu'inversement `EM` et `STRONG`

attribuent *structurellement* une importance particulière à ce mot, qui se trouve de ce fait affiché visuellement en italique ou en gras.

Comme cela a déjà été largement souligné, tous les effets « cosmétiques » appliqués à la présentation gagnent largement à être du ressort exclusif des feuilles de style.

Éléments ABBR et ACRONYM

Les éléments `ABBR` et `ACRONYM` permettent aux auteurs d'indiquer clairement les occurrences des abréviations et des acronymes. Les langues occidentales raffolent en effet des sigles comme RATP, ONU ou bit, ainsi que des abréviations comme Mme, SA ou etc. Le chinois et le japonais emploient tous deux des mécanismes d'abréviation analogues, selon lesquels un nom long est désigné par la suite par un sous-ensemble des caractères Han qui le constituent. Le balisage de ces structures fournit une aide précieuse aux agents utilisateurs et aux outils tels que les vérificateurs d'orthographe, les synthétiseurs de parole, les systèmes de traduction automatique et les moteurs de recherche.

Le contenu des éléments `ABBR` et `ACRONYM` spécifie l'expression abrégée telle qu'elle apparaîtrait normalement dans le cours du texte. L'attribut `title` de ces éléments peut servir à procurer la forme complète ou développée de l'expression.

Voici quelques exemples d'utilisation des éléments `ACRONYM` et `ABBR` :

Listing 5-3 : Éléments ABBR et ACRONYM

```
<P>
  <ABBR title="World Wide Web">WWW</ABBR>
  <ACRONYM lang="fr"
        title="Régie Autonome des Transports Parisiens">
     RATP
  </ACRONYM>
  <ABBR lang="es" title="Doña">Doña</ABBR>
  <ABBR lang="en" title="Abbreviation">abbr.</ABBR>
</P>
```

Voici l'aspect de ce fragment de code dans un navigateur. Remarquez ce qui se passe lorsque le pointeur est placé sur un élément (voir Figure 5.12).

Abréviations et acronymes possèdent souvent une prononciation idiosyncrasique. Par exemple, si CEE et FAI se prononcent typiquement

lettre à lettre, Ram et Medef se prononcent phonétiquement. En outre, certains termes (comme URI) sont épelés par les uns et prononcés comme des mots par les autres. En cas de nécessité, les auteurs devront utiliser les feuilles de style pour spécifier la prononciation d'une forme abrégée.



Figure 5.12 :
Éléments ABBR et ACRONYM

Éléments CODE, SMP et KBD

Internet possède, de par ses origines militaires et universitaires, une lourde orientation scientifique et technologique. Il était donc logique que HTML propose un élément structurel spécifique à l'affichage de code, l'élément `CODE`.

Au sens le plus large, le terme « code » décrit les éléments et la syntaxe permettant à une machine de lire un programme ou un fichier. En rédigeant de l'HTML, vous écrivez du code, même si celui-ci diffère largement de celui d'un programme écrit dans un autre langage, comme C ou PHP. Tout code doit respecter scrupuleusement une syntaxe spécifique au langage concerné. Traditionnellement (et comme c'est le cas dans cet ouvrage) du code est le plus souvent affiché sous forme de texte avec une police non proportionnelle. C'est une méthode universelle qui signale qu'il s'agit de quelque chose destinée à la machine. Lorsqu'il est lu par un agent utilisateur visuel, comme un navigateur, le texte figurant dans l'élément `CODE` devrait également être affiché à l'aide d'une police non proportionnelle. Certains navigateurs étant réglés afin d'afficher le contenu de l'élément `CODE` d'une façon particulière, mieux vaut le réserver aux exemples de code que vous souhaitez présenter sur le Web.

Voici un exemple de code, tiré de l'excellent ouvrage *La Bible C++* de C. Horstmann et T. Budd (Ed. Micro Application, 2004, ISBN 2-7429-3717-x).

Listing 5-4 : Exemple de l'élément CODE

```
<P>Fichier temps1.cpp</P>
<HR>
<P><CODE>
 1    #include &lt;iostream&gt;<BR>
 2   <BR>
 3    using namespace std; <BR>
 4   <BR>
 5    #include "ccc_time.h"<BR>
 6    <BR>
 7    int main()<BR>
 8    {<BR>
 9       Time reveil(7, 0, 0, 0); <BR>
10       reveil.add_seconds(1000); /* mille secondes plus tard */<BR>
11       cout &lt;&lt; reveil.get_hours()<BR>
12          &lt;&lt; ":" &lt;< reveil.get_minutes()<BR>
13          &lt;&lt; ":" &lt;&lt; reveil.get_seconds() &lt;&lt; "\n";<BR>
14    <BR>
15       return 0; <BR>
16    }
</CODE></P>
```

Voici l'aspect de ce code dans un navigateur.

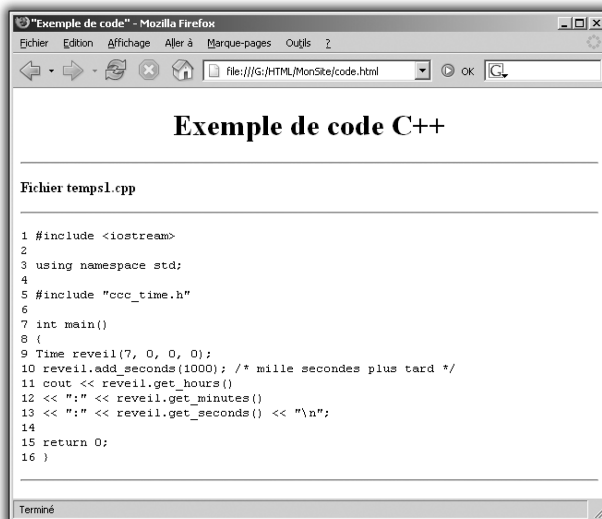


Figure 5.13 : *Mise en œuvre de l'élément CODE*



Caractères spéciaux

En examinant le code ci-dessus, vous devriez être intrigué par les mystérieux `<`. Plusieurs caractères sont des composants fondamentaux de la syntaxe HTML : en particulier les symboles <> qui constituent une balise. Il est donc impossible d'employer ces symboles en HTML sans que l'agent utilisateur ne tente de lire la balise placée à l'intérieur. Pour pallier cela, une notation particulière existe, nommée *référence de caractère*. Ce sujet sera abordé un peu plus loin dans ce chapitre.

L'élément `CODE` est de type ligne. Employez-le avec un élément de paragraphe, comme `P` ou `PRE`, pour afficher un bloc de code. Comme il s'agit d'un élément de structure significatif, ne placez que du code dans un élément `CODE`. Si vous souhaitez afficher du texte avec le même aspect, préférez l'élément bloc `PRE` ou l'élément ligne `TT` : tous deux affichent en principe un texte dans une police non proportionnelle.

L'élément `SMP`, très proche de l'élément `CODE`, sert à signaler une sortie de programme. En reprenant l'exemple précédant et en ajoutant à la fin, après la balise de fermeture de l'élément `P` :

```
<P>Ce programme affiche :</P>
<P><SMP>7:16:40</SMP></P>
```

le navigateur affiche :

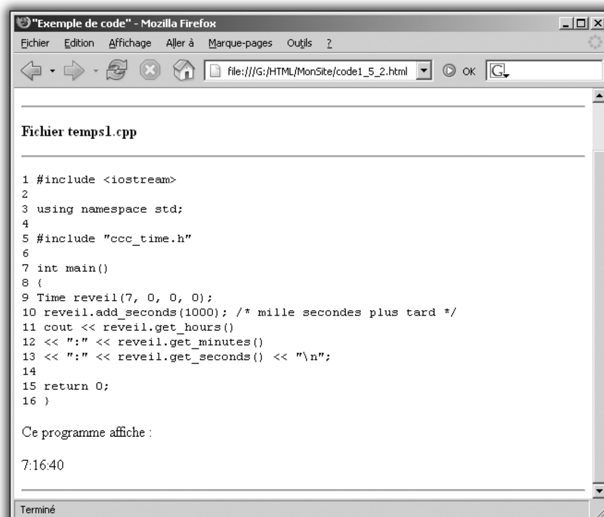


Figure 5.14 : *Mise en œuvre des éléments CODE et SMP*

L'élément KBD est très similaire : il sert à signaler un texte, ou une expression, devant être saisi par l'utilisateur au clavier.

Autres éléments structuraux

L'élément CITE a déjà été abordé lors de l'étude des éléments BLOCKQUOTE et Q. Il permet de fournir la source d'une citation et est généralement à préférer à l'attribut cite des éléments BLOCKQUOTE et Q.

Les autres éléments, DFN et VAR, sont plus rarement mis en œuvre.

Le tableau suivant présente tous ces éléments structuraux ainsi que leur signification.

Tableau 5.1 : *Éléments de structuration du texte*

Élément	Description
EM	Indique une mise en emphase.
STRONG	Indique une emphase plus forte.
CITE	Contient un extrait ou une référence vers une autre source.
DFN	Indique qu'il s'agit de l'instance définissante du terme englobé.
CODE	Désigne un fragment de code informatique.
SAMP	Désigne un exemple des sorties d'un programme, d'un script, etc.
KBD	Indique un texte que doit saisir l'utilisateur.
VAR	Indique l'instance d'une variable ou le paramètre d'un programme.
ABBR	Indique une forme abrégée (par exemple WWW, HTTP, i.e., etc.).
ACRONYM	Indique un acronyme (par exemple, radar, Lan, etc.).

Élément OBJECT

Les versions antérieures de HTML permettaient aux auteurs d'inclure des images (grâce à l'élément IMG) et des applets (grâce à l'élément APPLET). Ces éléments souffraient de plusieurs contraintes, dont la

principale était probablement l'incapacité à inclure les types de médias nouveaux et à venir.

HTML 4 a donc introduit l'élément `OBJECT`, qui offre une solution générale aux inclusions d'objets génériques. Cet élément permet la spécification de tout ce dont a besoin l'objet pour sa présentation par un agent utilisateur : le code source, les valeurs initiales et les données d'exécution. Dans cette spécification, le terme « objet » désigne les contenus placés dans des documents HTML (applets, modules complémentaires ou *plug-ins*, gestionnaires de médias, etc.).

Ce nouvel élément `OBJECT` prend ainsi en charge quelques-unes des tâches effectuées par les éléments existants. Il remplace notamment les éléments spécifiques `IMG` (image), `APPLET` (applet) et `IFRAME` (autre document HTML).

Chacun de ces trois types d'inclusion possède ainsi une solution spécifique et une solution générique. L'élément générique `OBJECT` servira de solution pour l'implémentation des types de médias futurs.

Nous examinerons plus en détail cet élément lors de l'examen des solutions spécifiques d'inclusion, mais il méritait d'être cité dès à présent.

5.3. Caractères spéciaux et encodage de caractères

Si vos pages Web sont exclusivement destinées à un public occidental et sont rédigées uniquement en anglais, vous ne devriez guère rencontrer de problèmes de caractères, sauf pour l'affichage des caractères HTML réservés, qui pourraient être présents dans du texte : les réglages par défaut sont parfaits.

En revanche, si vous devez placer sur tout ou partie d'une page du texte en russe, hébreu ou chinois, ou même simplement en français, vous devez savoir comment employer les caractères spéciaux et les jeux de caractères.

Encodages de caractères

Les jeux de caractères, également nommés *encodages de caractères* couramment utilisés sur le Web comprennent ISO-8859-1 (appelé aussi *Latin-1*, utilisable pour la plupart des langues d'Europe de l'Ouest), ISO-8859-5 (qui gère le cyrillique), SHIFT_JIS (un encodage du japonais), EUC-JP (un autre encodage du japonais) et UTF-8 (un encodage ISO 10646 qui a recours à un nombre différent d'octets pour différents caractères). Les noms des jeux de caractères sont insensibles à la casse : `SHIFT_JIS`, `Shift_JIS`, et `shift_jis` sont équivalents.

La spécification HTML 4.0.1 n'indique pas quels encodages de caractères doit reconnaître l'agent utilisateur.

Pour reconnaître l'encodage de caractères employé, le moyen le plus simple consiste à employer le paramètre `charset` dans le champ `Content–Type` de l'en-tête du protocole HTTP. Ce protocole mentionne ISO-8859-1 comme encodage par défaut lorsque le paramètre `charset` est absent du champ `Content–Type` de l'en-tête. Cette recommandation s'est toutefois révélée inutile : certains serveurs ne permettent pas d'envoyer un paramètre `charset` et d'autres peuvent ne pas être configurés pour l'envoyer. Les agents utilisateurs ne doivent donc pas supposer une valeur par défaut pour le paramètre `charset`.

Pour éviter tout problème, les documents HTML peuvent (et devraient) comporter des informations explicites sur l'encodage de caractères du document, fournies à l'agent utilisateur à l'aide d'un élément `META`.

Ainsi, pour spécifier comme encodage de caractères du document EUC-JP, le document devrait inclure la déclaration `META` suivante :

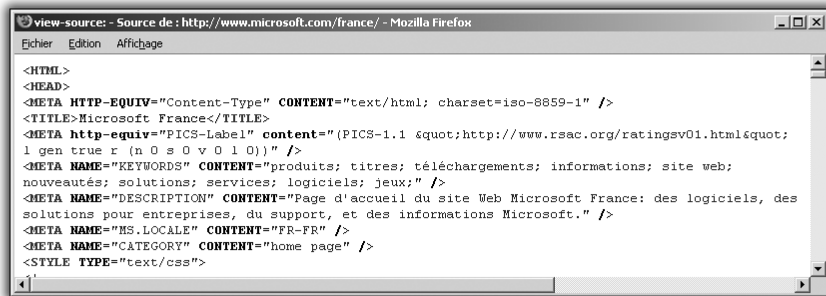
```
<META http-equiv="Content-Type" content="text/html;
charset=EUC-JP">
```

La déclaration `META` doit seulement être utilisée lorsque l'encodage est organisé de telle sorte que les octets en valeurs ASCII représentent des caractères ASCII (au moins jusqu'à l'analyse de l'élément `META`). Les déclarations `META` devraient donc apparaître le plus tôt possible dans l'élément `HEAD`.

C'est ce que respecte par exemple Microsoft (ainsi que de nombreux autres sites) en l'indiquant explicitement à l'aide d'un élément `META` placé au début du code source.



Figure 5.15 : *Page d'accueil du site Microsoft France (www.microsoft.com/france/)*


view-source: - Source de : http://www.microsoft.com/france/ - Mozilla Firefox

Fichier Edition Affichage

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1" />
<TITLE>Microsoft France</TITLE>
<META http-equiv="PICS-Label" content="(PICS-1.1 &quot;http://www.rsac.org/ratingsv01.html&quot;
l gen true r (n 0 s 0 v 0 l 0))" />
<META NAME="KEYWORDS" CONTENT="produits; titres; téléchargements; informations; site web;
nouveautés; solutions; services; logiciels; jeux;" />
<META NAME="DESCRIPTION" CONTENT="Page d'accueil du site Web Microsoft France: des logiciels, des
solutions pour entreprises, du support, et des informations Microsoft." />
<META NAME="MS.LOCALE" CONTENT="FR-FR" />
<META NAME="CATEGORY" CONTENT="home page" />
<STYLE TYPE="text/css">
```

Figure 5.16 : *Début du code source de la page d'accueil du site Microsoft France*

L'élément `META` employé par Microsoft est le suivant :

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=iso-8859-1" />
```

Remarquez tout d'abord qu'il est au format XHTML, comme l'indique sa fermeture `/>`. Cet élément indique que le contenu est de type HTML et que le jeu de caractères employé est ISO-8859-1, le jeu de caractères du monde occidental. Employer cet élément étant une bonne habitude à prendre, faites-le immédiatement pour la page d'accueil de votre site.

1. Ouvrez dans le Bloc-Notes le fichier *pageacc1_5_5.html*, en principe celui actuellement ouvert dans le Bloc-Notes.
2. Modifiez à nouveau le numéro de version :

```
<META name="version" content="1.5.6">
```

3 Ajoutez, juste après l'élément `TITLE`, un nouvel élément `META` :

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=iso-8859-1">
```

4 Enregistrez votre fichier sous le nom *pageacc1_5_6.html*.

Si votre page emploie un jeu de caractères différent, vous pouvez spécifier une autre valeur pour l'attribut `charset`. Cet attribut peut concerner toute la page ou s'appliquer uniquement à un élément (par exemple, pour une citation).

Caractères spéciaux

Un encodage de caractères donné peut ne pas pouvoir exprimer tous les caractères du jeu de caractères du document. Avec de tels encodages, ou lorsque les configurations matérielles ou logicielles ne permettent pas aux utilisateurs d'entrer directement certains caractères spéciaux, les auteurs peuvent avoir recours aux références de caractères SGML. Il s'agit d'un mécanisme indépendant de l'encodage, utilisé pour ajouter n'importe quel caractère issu du jeu de caractères du document.

Les références de caractères en HTML peuvent se présenter sous deux formes :

- Les références de caractères numériques (décimales ou hexadécimales).
- Les références d'entités de caractères.

Les *références de caractères numériques* spécifient la position du code d'un caractère dans le jeu de caractères du document. Elles peuvent être exprimées :

- Sous forme décimale, `&#D;`, où `D` représente un nombre décimal qui se rapporte au numéro de caractère `D` décimal ISO 10646.
- Sous forme hexadécimale, `&#xH;` où `&#XH;`, où `H` représente un nombre hexadécimal qui se rapporte au numéro de caractère `H` hexadécimal ISO 10646.

Les nombres hexadécimaux dans les références de caractères numériques sont insensibles à la casse. Il est recommandé de préférer la forme hexadécimale à la forme décimale, car c'est la forme employée pour les normes des jeux de caractères.

Les *entités de caractères* représentent un moyen plus intuitif pour les auteurs d'appeler les caractères du jeu de caractères du document. Elles ont recours à des noms symboliques, qui évitent aux auteurs d'avoir à se remémorer les positions des codes. Par exemple, la référence d'entité de caractère `à` se rapporte à la lettre « `à` ». Il est plus facile de se souvenir de `à` que de `à` ou de `à`.

HTML 4 ne définit pas de référence d'entité de caractère pour chacun des caractères du jeu de caractères du document. En outre, les références d'entités de caractères sont sensibles à la casse. Ainsi, `À` (`À`) ne se rapporte pas au même caractère que `à` (`à`).

Quatre références d'entités de caractères méritent une attention particulière : elles sont fréquemment utilisées pour masquer certains caractères spéciaux dans le code HTML.

- `<` représente le caractère <
- `égt;` représente le caractère >
- `&` représente le caractère `&`
- `"` représente le caractère `"`

Vous comprenez désormais l'origine des caractères « bizarres » que vous avez rencontrés dans le passé. Vous pourriez cependant vous poser la question de l'intérêt réel de ces références de caractères : si elles paraissent effectivement indispensables pour les caractères réservés de HTML, comme < et >, nous avons jusque-là largement employé, dans les pages, du texte comportant des caractères accentués sans que cela ne semble poser le moindre problème.

La restitution des caractères accentués présents tels quels dans le texte dépend en réalité tant de l'agent utilisateur employé que de la configuration de la machine utilisée, et notamment des polices installées sur celle-ci. Par sécurité, l'emploi des références de caractères est donc hautement recommandé.

La majorité des éditeurs HTML transforment automatiquement ces caractères en références, vous évitant d'avoir à les saisir manuellement : ainsi, `Liste ordonnée` devient `Liste ordonna#xe9;e`, mais vous pourriez aussi bien écrire `Liste ordonnée` ou `Liste ordonnée`. Si vous préférez vous en tenir à un éditeur de type texte, vous devrez en tenir compte et « *vous habituer à*

écrire vos textes en tenant compte des caractères accentu&eacutes;s ». Rassurez-vous, l'habitude vient en fait assez vite.

1 Ouvrez dans le Bloc-Notes le fichier *pageacc1_5_6.html* (en principe celui actuellement ouvert dans le Bloc-Notes).

2 Modifiez à nouveau le numéro de version :

```
<META name="version" content="1.5.7">
```

3 Remplacez tous les caractères accentués du texte par leurs équivalents références de caractères :

```
<LI>sur ma <A href="region.html">r&eacute;gion</A> ;</LI>
…
<ADDRESS>
<A href="../ORignal/">Olivier Rignal</A> et
<A href="../JBogue/">Justin Bogue</A> sont les personnes
&agrave contacter pour tout <A href="probleme">
probl&egrave;me</A> rencontr&eacute; sur le site.
</ADDRESS>
```

4 Enregistrez votre fichier sous le nom *pageacc1_5_7.html*.

Examinez cette page dans votre navigateur : vous ne devriez constater aucune différence avec la version précédente.

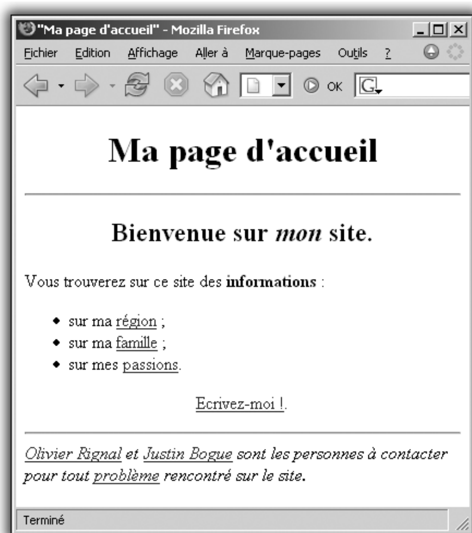


Figure 5.17 : *Emploi des références de caractères*

Un peu pénible quand même… Ne serait-il pas possible de tirer profit de la capacité des éditeurs HTML à transformer automatiquement tout ce qui a pu être créé auparavant ? Bien sûr que oui.

1. Ouvrez Nvu, puis ouvrez le fichier *pageacc1_5_6.html* (qui possède encore des caractères accentués).
2. Vérifiez le paramètrage de Nvu. Choisissez **Outils** > **Préférences**. Dans la fenêtre qui s'ouvre, cliquez dans le volet de droite sur **Général** si le volet **Général** n'est pas affiché dans la partie droite. Vérifiez que dans la section *Lors de l'enregistrement ou de la publication de page* l'option *Reformater la source HTML* est bien sélectionnée.
3. Cliquez dans le volet de gauche sur **Avancées**. Dans la section *Caractères spéciaux*, vérifiez que *Définir les caractères suivants en tant qu'entités* est bien fixé à *Tout caractère ayant une entité équivalente dans la spécification HTML 4*".
4. Enregistrez votre fichier sous le nom *pageacc1_5_6b.html*.

En examinant le code source, vous constaterez que tous les caractères possédant une entité équivalente ont bien été remplacés par celle-ci.

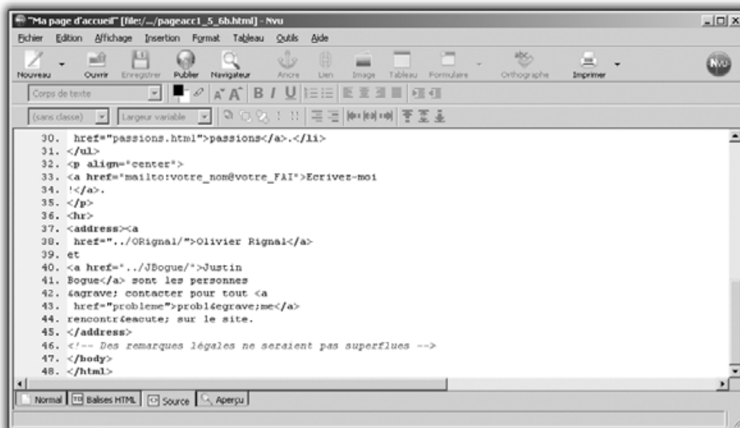


Figure 5.18 : *Page automatiquement transformée par Nvu*

Remarquez que, si nous avons ici modifié le nom du fichier, cela n'est pas obligatoire : le simple fait d'enregistrer le fichier accomplit la transformation.

Caractères non affichables

Malgré toutes les précautions, un agent utilisateur n'est pas toujours en mesure de restituer de manière significative tous les caractères d'un document. Par exemple, si l'agent utilisateur ne dispose pas d'une police appropriée, la valeur d'un caractère peut ne pas s'exprimer au travers de l'encodage de caractères interne de l'agent utilisateur, etc.

Vu la palette des réponses possibles en pareils cas, la spécification HTML 4.0.1 ne suggère aucun comportement spécifique. Selon la mise en œuvre, les caractères non affichables peuvent également être gérés par le système d'affichage sous-jacent et non par l'application elle-même. Il est recommandé par le W3C que les agents utilisateurs adoptent un mécanisme visible et clair, mais non intrusif, pour avertir l'utilisateur de l'absence des ressources nécessaires.

Le tableau suivant présente les entités les plus employées en HTML.

Tableau 5.2 : *Jeu des références de caractères Latin-1 (ISO-8859-1)*

Caractère	Entité	Décimal	Hexadécimal	Caractère	Entité	Décimal	Hexadécimal
<	<	<	<	é	é	é	é
>	>	>	>	ê	ê	ê	ຬ
espace insécable				ì	ì	ì	ì
«	«	«	«	î	î	î	î
±	±	±	±	ï	ï	ï	ï
»	»	»	»	ô	ô	ô	ô
×	×	×	·	ö	ö	ö	ö
à	à	à	à	÷	÷	÷	÷
â	â	â	â	ù	ù	ù	ù
ç	ç	ç	ç	û	û	û	û
è	è	è	è	ü	ü	ü	ü

L'Annexe D présente une liste plus complète des caractères du jeu Latin-1.

Nous avons déjà rencontré avec les tableaux l'attribut `dir`, définissant la directionnalité. Celui-ci, combiné ou non avec l'attribut `lang`, permet d'adapter encore mieux vos pages au public visé. Ces attributs seront étudiés plus en détail dans le Chapitre 10, traitant de l'internationalisation et de la localisation.

5.4. Modification de l'apparence du texte

HTML dispose de plusieurs éléments permettant de modifier l'apparence du texte, similaires en un sens aux enrichissements que vous pouvez apporter à un document dans un traitement de texte. Il est ainsi possible de modifier le style d'un mot individuel, d'une expression ou d'une phrase. Il existe de bonnes raisons de vouloir procéder ainsi, mais il existe également d'excellentes raisons de ne pas le faire. Même si tous ces éléments ne sont pas déconseillés, il reste préconisé de leur préférer les feuilles de style.

Éléments TT, I, B, BIG, SMALL, STRIKE, S et U

Ajouter du gras, de l'italique et du soulignement à certaines parties de vos textes augmente leur impact et contribue à l'efficacité de votre message pour le public concerné. Comme cela a déjà été signalé, mieux vaut toutefois employer une emphase structurelle à l'aide des éléments `EM` et `STRONG`, déjà abordés.

Le tableau suivant présente les éléments utilisables.

Tableau 5.3 : *Éléments de modification de police*

Élément	But	Exemple
TT	Texte affiché en police non proportionnelle	<TT>texte</TT>
I	Texte en italique	<I>texte</I>
B	Texte en gras	<B>texte</B>
BIG	Texte affiché dans une police plus grande	<BIG>texte</BIG>

Tableau 5.3 : *Éléments de modification de police*		
Élément	**But**	**Exemple**
SMALL	Texte affiché dans une police plus petite	<SMALL>texte</SMALL>
STRIKE et S	Texte barré (déconseillé)	<S>texte</S>
U	Texte souligné (déconseillé)	<U>texte</U>

Remarquez que la restitution des éléments de style de police dépend de l'agent utilisateur.

Il est possible d'obtenir une bien plus grande diversité d'effets de police en utilisant les feuilles de style. Les éléments de style de police doivent être correctement imbriqués. La restitution des éléments de style de police imbriqués dépend de l'agent utilisateur.

Modificateurs de police : éléments FONT et BASEFONT

L'élément BASEFONT définit les caractéristiques de la police de base. Il est généralement placé en tête d'un document HTML. Les modifications ultérieures apportées à la police grâce à l'élément FONT sont relatives à la police de base définie par l'élément BASEFONT.

Ces deux éléments sont désormais déconseillés. Ils possèdent plusieurs attributs facultatifs :

Attribut size

Cet attribut déconseillé définit la taille de la police. Sa valeur peut être exprimée comme entier compris 1 et 7, attribuant à la police une certaine taille fixe dont la restitution dépend de l'agent utilisateur, ou comme accroissement relatif : la valeur +1 signifie la taille au-dessus, la valeur –3 signifie trois tailles en dessous. Toutes les tailles se trouvent sur une échelle allant de 1 à 7.

Si l'élément BASEFONT est absent ou ne définit pas l'attribut size, la taille de police de base est égale à 3.



Police des titres

La taille de police de base ne s'applique pas aux titres, sauf quand ceux-ci sont modifiés par un élément `FONT` avec un changement de taille de police relatif.

Attribut color

Cet attribut déconseillé spécifie la couleur du texte (reportez-vous au Chapitre 6 pour plus de détails).

Attribut face

Cet attribut définit une liste de noms de police, séparés par des virgules, que l'agent utilisateur devrait employer par ordre de préférence. Vous trouverez ci-dessous un petit exemple de la mise en œuvre de cet attribut. L'affichage dans un navigateur est montré dans la figure qui suit.

Listing 5-5 : Tailles de police

```
<P align="center"><font size=1>size=1</font><BR>
<font size=2>size=2</font><BR>
<font size=3>size=3</font><BR>
<font size=4>size=4</font><BR>
<font size=5>size=5</font><BR>
<font size=6>size=6</font><BR>
<font size=7>size=7</font></P>
```



Figure 5.19 :
Différents types de police

Une parade aux nombreux problèmes posés par l'élément `FONT` consiste à recourir aux feuilles de style en cascade (CSS). Lorsqu'elles sont utilisées correctement, les feuilles de style permettent aux concepteurs Web de suggérer un format d'affichage pour certaines portions de texte, sachant que le navigateur de l'utilisateur peut ne pas en tenir compte. Vous pouvez modifier la couleur, la taille, la graisse et le style du texte. Vous pouvez

même disposer d'un style de recouvrement et de justification de paragraphe. Cette méthode, bien plus souple que les éléments disponibles *via* HTML, est étudiée en détail dans le Chapitre 8. En fait, les plus anciens navigateurs (ceux qui n'utilisent pas CSS) ignorent automatiquement les feuilles de style, ramenant tout dans le style de texte par défaut.



Tailles relatives

Si vous décidez d'avoir recours à l'élément `FONT` pour modifier la taille ou les couleurs d'une page fondamentale, servez-vous de tailles relatives (`size=+1` au lieu de `size=4`) et choisissez des couleurs ressortant sur un fond blanc, noir ou gris, au cas où le fond serait ainsi prédéfini dans les préférences du navigateur.

Indices et exposants : éléments SUB et SUP

Le français emploie régulièrement des abréviations dans lesquelles un des termes est écrit comme exposant : Mlle, 1er, etc. En outre, les notations mathématiques (étudiées un peu plus loin) ont fréquemment recours à des exposants : $a^2 + b^2 = c^2$.

HTML propose à cette fin les éléments `SUB` (*subscript*, indice) et `SUP` (*superscript*, exposant). Il s'agit d'éléments ligne qui décalent légèrement vers le bas ou vers le haut le texte par rapport au reste de la ligne. Leurs balises d'ouverture et de fermeture sont obligatoires.

L'élément indice `SUB` place le texte *en dessous* de la ligne de texte normal. Cela est fréquemment employé pour des notes de bas de page ou de glossaire, souvent liées à l'aide d'un signet (ou ancre) vers une référence située à la fin du document. Par exemple, considérez le fragment de code suivant :

```
<P>Il existe d'ailleurs de nombreux widgets<SUB>
<A HREF=#note1>[1]</A></SUB> compl&eacute;mentaires
Disponibles en t&eacute;l&eacute;chargement...</P>
<P><BR><BR><BR><BR><BR></P>
<P>... suite du contenu ...</P>
<P><BR><BR><BR><BR><BR></P>
<P><CITE id="note1">[1] Widget (<B>WI</B>ndows ga<B>DGET</B>).
 D&eacute;signe sous Unix et Linux un outil permettant
 d'accomplir une action. Plus connu sous le nom de
 "contr&ocirc;le" avec Windows.</CITE></P>
```

Ce code procure un lien hypertexte vers la définition du terme *widget*.

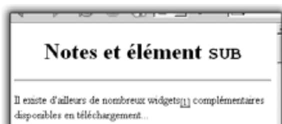


Figure 5.20 :
Le premier paragraphe présente un lien

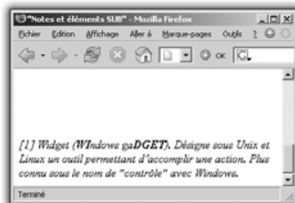


Figure 5.21 :
Le premier paragraphe présente un lien. Un clic sur celui-ci mène vers la définition du terme.

Comme vous l'avez sans doute deviné tout seul, l'élément exposant accomplit exactement le contraire, plaçant le texte au-dessus de la ligne de texte normal. Bien que cela puisse également être utilisé pour des notes, l'usage le plus fréquent concerne des notations chimiques ou mathématiques. Ainsi,

```
a<SUP>2</SUP> + b<SUP>2</SUP> = c<SUP>2</SUP>
```

affiche à l'écran : $a^2 + b^2 = c^2$

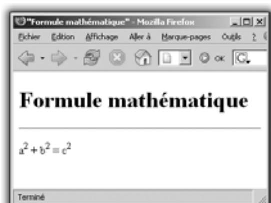


Figure 5.22 :
Équation obtenue à l'aide de l'élément SUP

Remarquez que nous avons employé ici un élément CODE pour afficher l'équation dans une police non proportionnelle.

Les termes liés à la langue, les notes, les équations chimiques ou mathématiques sont autant de raisons en faveur d'un emploi judicieux des indices et des exposants.



Emploi intelligent des notes en HTML

Une note est liée au corps d'une section et expliquée soit à la fin de cette section soit à la fin du document. Comme certains documents peuvent être de grande taille, préférez en HTML les notes placées en fin de section ou en pied de page, pour épargner à vos utilisateurs une navigation intensive.

5.5. Notation mathématique

Comme nous l'avons déjà souligné, le World Wide Web a vu le jour en 1991, avant tout pour favoriser le transfert de données entre les physiciens. Malgré cette origine, les notations scientifiques et mathématiques étaient mal prises en charge par les premières versions de HTML. HTML 4.0 propose cependant quelques méthodes très sophistiquées pour afficher de façon attrayante et fonctionnelle les équations mathématiques. De plus, divers programmes auxiliaires offrent d'autres moyens d'afficher des mathématiques sur le Web.

En texte simple

La façon la plus simple d'afficher des mathématiques sur le Web est parfois la meilleure, même si ce n'est pas la plus attrayante : vous pouvez recourir à des caractères textuels pour les représenter, comme cela est montré dans l'exemple de l'élément `SUP`. Comme nous l'avons fait dans cet exemple, mieux vaut recourir à un élément `CODE` pour afficher l'équation dans une police non proportionnelle (à espacement fixe).

Vous devrez cependant parfois employer dans des équations des symboles particuliers, comme des lettres grecques. Dans un tel cas, l'emploi de l'élément `FONT` déjà présenté, bien que déconseillé, reste une solution :

Listing 5-6 : Notations scientifiques en texte

```
<H1>Formules statistiques</H1>
<HR>
<P>Intervalle de confiance de la moyenne <I>m</I> pour
une population normale d'&eacute;cart-type
<FONT face="Symbol">s</FONT> à échantillon <I>n</I> :<BR><BR>
<CODE><BIG><BIG>
[x - t &times; (<FONT face="Symbol">s</FONT>/n<SUP>&frac12;</SUP>) ;
x + t &times; (<FONT face="Symbol">s</FONT>/n<SUP>&frac12;</SUP>)]
</CODE></BIG></BIG>
</P>
```

Hélas ! Tous les navigateurs n'interprètent pas ce code de la même façon. Seul Internet Explorer tire profit de l'élément `FONT` pour afficher correctement le sigma grec.

Intervalle de confiance de la moyenne *m* pour une population normale d'écart-type σ à échantillon *n* :

$[x - t \times (\sigma/n^{½}) \; ; \; x + t \times (\sigma/n^{½})]$

Figure 5.23 : *Exemple d'équation écrite à l'aide de caractères texte avec Internet Explorer*

Intervalle de confiance de la moyenne *m* pour une population normale d'écart-type s à échantillon *n* :

$[x - t \times (s/n^{½}) \; ; \; x + t \times (s/n^{½})]$

Figure 5.24 : *Exemple d'équation écrite à l'aide de caractères texte avec Firefox*

Intervalle de confiance de la moyenne *m* pour une population normale d'écart-type s à échantillon *n* :

$[x - t \times (s/n^{½}) \; ; \; x + t \times (s/n^{½})]$

Figure 5.25 : *Exemple d'équation écrite à l'aide de caractères texte avec Amaya.*

L'emploi de texte simple atteint toutefois ses limites en présence de fractions compliquées, ou en cas d'opérandes mathématiques plus avancés.

À l'aide de graphismes

Une autre option consiste à créer des images de l'équation mathématique et de les insérer sur votre page, comme vous le feriez pour tout graphisme (ce sujet sera traité dans le Chapitre 7). Vous pouvez pour ce faire recourir à un éditeur mathématique comme l'Éditeur d'équation de Microsoft Word ou un programme commercial comme MathType (**www.dessci.com/en/products/mathtype/**).

Utiliser des fichiers graphiques séparés est une solution simple et rapide, mais les images ne peuvent pas être facilement enregistrées avec les pages Web, tandis qu'il reste impossible de couper/coller les équations. Créer une image individuelle pour chaque équation peut représenter un travail titanesque si la page traite intensivement des mathématiques.

L'apparence finale peut également être variable selon les paramétrages définis pour la page Web et pouvant être outrepassés par l'utilisateur.

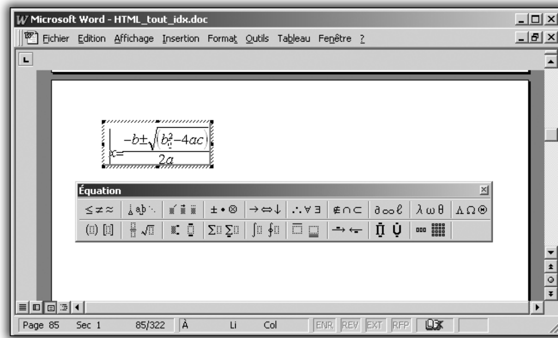


Figure 5.26 : *Exemple d'emploi de l'éditeur d'équation Word*

À l'aide de MathML

Différents experts du Web, mathématiciens, scientifiques et autres ont travaillé pour développer une solution qui permette d'incorporer les notations mathématiques en HTML. C'est un problème complexe dont le but était de définir un langage de notation structurel plutôt qu'un simple langage de description de page. HTML 4.0 propose une excellente solution, sous la forme de MathML, un langage de notation mathématique développé à partir de XML. La spécification 1.01 de MathML est disponible à l'adresse **www.w3.org/1999/07/REC-MathML-19990707/**

MathML est simple, bien que par nécessité complexe pour pouvoir représenter la totalité des mathématiques modernes. Comme en matière de conception Web, il existe différents éditeurs, commerciaux ou gratuits, qui permettent de créer des équations grâce à une interface simple et facile d'accès, de convertir votre travail, et de laisser au programme la tâche de résoudre votre équation. Même si, comme en HTML, il est possible de coder directement en MathML, mieux vaut employer un éditeur. Nous ne saurions mieux vous conseiller que Amaya, cet excellent et universel produit gratuit développé par le W3C…

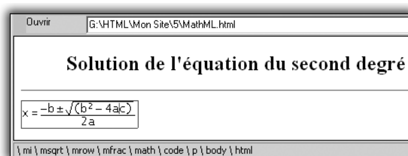


Figure 5.27 : *Amaya permet de réaliser très facilement des équations mathématiques sous MathML*

L’exemple présenté est la célèbre solution de l’équation du second degré. Remarquez le pointeur placé sur l’éditeur mathématique. Le code de cette page est présenté ci-dessous.

Listing 5-7 : Exemple MathML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;
   charset=UTF-8">
  <title>MathML</title>
  <meta name="generator"
        content="amaya 8.7.3, see http://www.w3.org/Amaya/">
</head>

<body>
<h1 align=center>Solution de l’&eacute;quation du second
 degr&eacute;</h1>
<p>
<code>
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mi style="font-style: normal">x</mi>

  <mo>=</mo>
  <mfrac>
    <mrow>
      <mo>-</mo>
      <mi style="font-style: normal">b</mi>
      <mo>±</mo>
      <msqrt>

        <mo>(</mo>
        <msup>
          <mi style="font-style: normal">b</mi>
          <mn>2</mn>
        </msup>
        <mo>-</mo>
        <mn>4</mn>
        <mi>ac</mi>
        <mo>)</mo>
      </msqrt>
    </mrow>
    <mrow>
      <mn>2</mn>
      <mi style="font-style: normal">a</mi>
    </mrow>
  </mfrac>
</math>
</code>
</p>
```

```
</body>
</html>
```

Malheureusement, rares sont les navigateurs à reconnaître MathML : pour le moment, Firefox et Internet Explorer en restent encore totalement incapables.



Figure 5.28 :
Contrairement à Amaya, ni Firefox 2.0



Figure 5.29 :
... ni Internet Explorer 7.0 ne savent actuellement reconnaître MathML

Cet état de choses devant logiquement évoluer, nous ne saurions faire autrement que de vous recommander de vous intéresser à MathML.

5.6. Résumé

- L'élément HR permet d'insérer une ligne horizontale à l'écran.
- La balise <!— commentaire —> signale un commentaire ignoré par HTML. Ces commentaires sont précieux lorsque le site est développé et/ou maintenu par plusieurs personnes.
- Les éléments `DIV` et `SPAN`, grâce à leurs attributs `id` et `class`, offrent un mécanisme générique de structure. Ils sont essentiellement employés avec les feuilles de style (grâce aux attributs `id` et `class`) et/ou l'attribut `lang` pour mieux exploiter HTML.
- L'élément `PRE` définit un bloc de texte préformaté : la mise en forme stipulée est théoriquement restituée telle quelle à l'écran, y compris les espaces vierges normalement ignorés par HTML.

- L'élément `BLOCKQUOTE` (de type bloc) et son équivalent ligne `Q` servent à signaler une citation. Ils doivent de préférence être employés en combinaison avec l'élément `CITE`, procurant la source de la citation.
- L'élément `ADDRESS`, généralement placé au début ou à la fin d'un document, sert à fournir des informations de contact applicables au document.
- L'attribut `style` permet de modifier les marges d'un bloc de texte.
- Les éléments `EM` et `STRONG` s'utilisent pour mettre en exergue une portion du texte, qui possède par exemple une signification particulière dans un contexte technique.
- `ABBR` et `ACRONYM` sont des éléments qui permettent aux auteurs d'indiquer clairement les occurrences des abréviations et des acronymes.
- L'élément `CODE` a pour but d'afficher un fragment de code informatique. C'est un élément structurel, accompagné des éléments `PMR` (sortie de programme) et `KBR` (texte à saisir au clavier).
- Le jeu de caractères habituellement employé en Europe occidentale est ISO-8859-1, également nommé Latin-1. Il existe toutefois de nombreux autres encodages de caractères répondant aux besoins des différentes langues.
- Les caractères ne pouvant être obtenus directement au clavier (comme les majuscules accentuées) ou susceptibles de poser problème dans du code HTML (comme les caractères < et >) sont remplacés dans le code HTML par des références de caractères, exprimées sous forme numérique ou à l'aide d'entités caractères.
- `FONT` et `BASEFONT` sont des éléments qui permettent de définir et de modifier les polices employées. Ils sont désormais déconseillés au profit des feuilles de style.
- Les indices et exposants sont obtenus à l'aide des éléments `SUB` et `SUP`.
- Les équations mathématiques peuvent être représentées à l'aide de texte brut, ou d'images réalisées à l'aide d'autres produits ou de MathML, la solution préconisée par le W3C pour l'affichage de notations mathématiques. MathML reste malheureusement très peu pris en charge par les navigateurs actuels.

Chapitre 6

Couleurs et images

Les couleurs 177
Les images 186
Images animées 198
Image cliquable 210
Inclusion générique d'images : élément OBJECT 224
Résumé 240

La plupart des pages Web ont recours à des couleurs, des images et des icônes exploitées d'une façon ou d'une autre. Choisir avec soin un ensemble d'images contribue à donner à votre site son style particulier. Cet ensemble procure aux visiteurs des indices visuels sur le contenu et est également susceptible de servir d'outil de navigation.

La première étape pour passer d'une simple page de texte en noir et blanc à une page plus attractive consiste à employer les couleurs. L'étape suivante consiste à incorporer des images en divers emplacements de la page : dans des éléments, comme arrière-plan d'une page ou d'une portion de celle-ci, etc.

En règle générale, ne demandez *jamais* à une image de faire le travail d'un texte. De simples liens texte peuvent conduire les visiteurs vers d'autres parties du site aussi facilement qu'un bouton graphique, sans augmenter le temps de chargement. Si vous retenez des boutons graphiques, ceux-ci doivent être aussi « légers » que possible.

Les images sont toutefois précieuses : vous pouvez employer le logo de votre organisme dans l'en-tête, ajouter d'autres images comme éléments illustratifs, et même employer un « texte icône » c'est-à-dire une police de caractères fortement associée à votre site.

Outre les images statiques, vous pouvez employer des images animées (gif animés). Comme vous le verrez dans ce chapitre, il faut dans ce domaine agir avec prudence : un grand nombre d'images animées peut fatiguer l'œil assez vite et rendre la page plus confuse qu'explicite.

Enfin, une étape encore plus évoluée consiste à créer des images représentant plusieurs liens : les images cliquables (*imagemap*). HTML permet de faire en sorte qu'une zone de l'image soit active, c'est-à-dire qu'un lien lui soit associé. Vous allez étudier ces différentes techniques dans ce chapitre.

Éléments étudiés :

```
bgcolor, text, link, vlink, alink, color
IMG, src, alt
AREA, shape, coords, nohref
MAP
usemap
OBJECT, data, type
```

6.1. Les couleurs

La toute première étape de l'enjolivement d'une page, après avoir parfaitement défini sa structure comme vous avez appris à le faire dans les chapitres précédents, consiste à y ajouter de la couleur.

Les couleurs sont souvent mises à profit par les concepteurs de sites pour donner à un site une note particulière, ce qui porte généralement le nom de « charte graphique ». Lorsque vous surfez sur le Web, essayez de noter les pages qui vous plaisent et celles que vous n'aimez pas, et pourquoi. Se servir de couleurs spécifiques sur l'ensemble du site aide les visiteurs à se souvenir de celui-ci, sans que le temps de chargement de la page soit augmenté.

Exploiter les couleurs est de la plus grande simplicité en HTML, grâce à divers attributs de l'élément `BODY` : `bgcolor` et `text` définissent respectivement la couleur d'arrière-plan et la couleur de texte, `link`, `vlink` et `alink` définissent respectivement la couleur des liens, la couleur des liens visités et la couleur des liens sélectionnés. `bgcolor` sert également à définir la couleur de fond dans des tableaux, tandis que l'attribut `color`, propre aux éléments `FONT` et `BASEFONT`, définit la couleur du texte dans un élément individuel.

La valeur des attributs de couleur se rapporte aux définitions des couleurs telles que spécifiées dans la spécification SRGB, relative à un standard des couleurs sur Internet et actuellement dans sa version 1.10 (www.w3.org/Graphics/Color/sRGB). Une valeur de couleur peut soit être un nombre hexadécimal (préfixé par un caractère dièse « `#` ») soit l'un des noms de couleur prédéfinis.

Tableau 6.1 : *Noms de quelques couleurs et valeurs RGB*

Nom	Valeur RGB	Nom	Valeur RGB
Black	`#000000`	Green	`#008000`
Silver	`#C0C0C0`	Lime	`#00FF00`
Gray	`#808080`	Olive	`#808000`
White	`#FFFFFF`	Yellow	`#FFFF00`
Maroon	`#800000`	Navy	`#000080`
Red	`#FF0000`	Blue	`#0000FF`

Tableau 6.1 : Noms de quelques couleurs et valeurs RGB			
Nom	**Valeur RGB**	**Nom**	**Valeur RGB**
Purple	`#800080`	Teal	`#008080`
Fuchsia	`#FF00FF`	Aqua	`#00FFFF`

Ces noms de couleur sont insensibles à la casse : les valeurs de couleur `#800080`, `Purple` et `PURPLE` désignent toutes trois la même couleur violette.

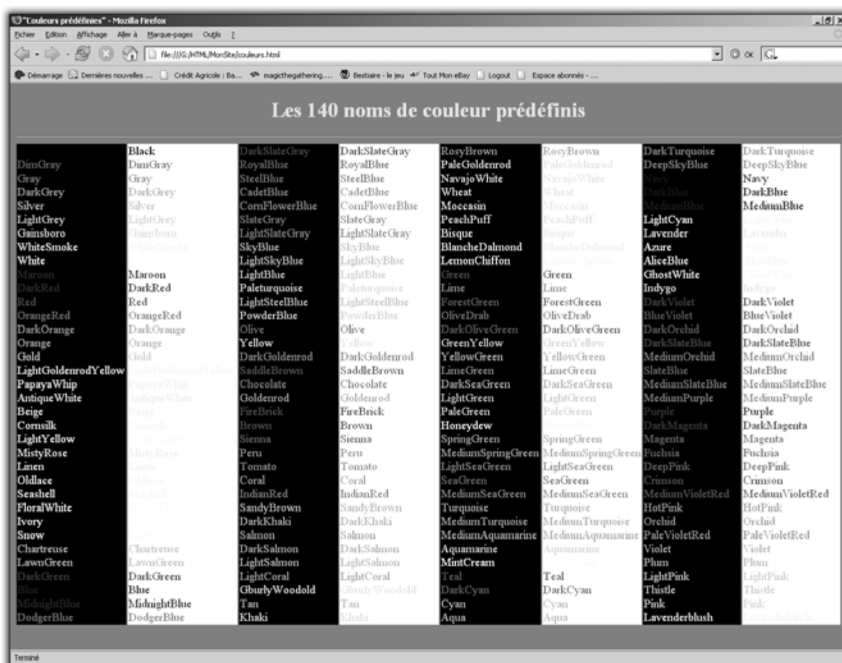


Figure 6.1 : *Les 140 couleurs à noms prédéfinis*

Modification du schéma de couleurs de la page

Modifier un schéma de couleur à l'intérieur d'une page s'effectue dans l'élément `BODY`, car cela affecte pratiquement tous les éléments du corps de la page. Après une modification de couleur, vérifiez-la toujours en la

visionnant avant de la publier sur le Web, si possible avec le maximum d'ordinateurs et de configurations différentes. Cela afin de bien vérifier que la page reste parfaite en toutes circonstances.

Nous allons modifier le schéma de couleur de notre page d'accueil, tout d'abord à l'aide des couleurs standard :

1 Ouvrez dans le Bloc-Notes le fichier *pageacc1_5_7.html* (la dernière version de la page d'accueil).
2 Modifiez à nouveau le numéro de version :

```
<META name="version" content="1.6.1">
```

3 Attribuez une couleur d'arrière-plan à l'intégralité de la page :

```
<BODY bgcolor="black">
```

4 Attribuez une couleur au texte de la page (si vous ne le faites pas, la couleur par défaut du texte étant le noir, toute la page sera invisible) :

```
<BODY bgcolor="black text="white">
```



BODY ou BASEFONT

Vous pouvez également modifier la couleur du texte à l'aide d'un élément BASEFONT. Un élément

```
<BASEFONT color="white">
```

aurait un effet identique à

```
<BODY text="white">
```

5 Modifiez les couleurs des liens :

```
<BODY bgcolor="black" text="white" link="yellow"
vlink="silver" alink="purple">
```

4 Enregistrez votre fichier sous le nom *pageacc1_6_1.html*.

Examinez cette nouvelle page dans votre navigateur. D'accord, elle n'offre rien d'exceptionnel et pourrait même être décrite comme un peu macabre, mais il ne s'agit que d'un exemple ! L'emploi des noms de couleur génériques est d'une grande simplicité et parfaitement explicite. Vous pouvez indifféremment employer le codage hexadécimal et écrire :

```
<BODY bgcolor="#000000" text="#FFFFFF" link="#FFFF00"
vlink="#C0C0C0" alink="#800080">
```

La valeur hexadécimale employée porte le nom de *code RGB* de cette couleur. Dans ce code à six chiffres, les deux premiers chiffres correspondent à la quantité de rouge (R, *red*), les deux suivants à la quantité de vert (G, *green*) et les deux derniers à la quantité de bleu (B, *blue*). Plus les différentes valeurs sont élevées et plus la couleur est lumineuse : trois fois `0` pour le noir et trois fois `FF` pour le blanc. Vous pouvez spécifier n'importe quelle valeur hexadécimale, même si elle ne correspond pas à une couleur standard. C'est ce qu'a fait Micro Application.

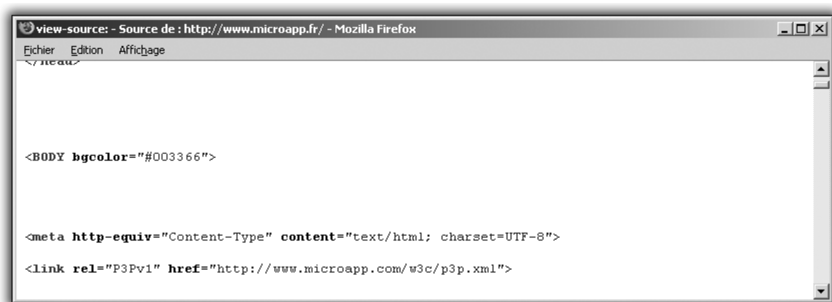


Figure 6.2 : *Micro Application a retenu comme couleur de fond un bleu profond, défini dans l'élément BODY grâce à l'attribut bgcolor="#003366".*

Comme vous êtes un utilisateur subtil, vous réalisez que cela permet de disposer de 256^3 couleurs possibles (de `0` à `FF` ou `255`, donc 256 possibilités pour chaque composante primaire), soit environ 16,8 millions de couleurs…

Vous risquez de dire alors : « *Rigolo, va ! Facile à dire, mais je n'ai pas envie de tester une multitude de combinaisons avant de trouver la couleur qui me plaît !* » Pas de panique, c'est enfantin. Même si vous ne disposez pas d'un logiciel de dessin évolué comme Adobe PhotoShop ou Corell Draw (ou, en étant moins dépensier, d'un partagiciel comme Paint Shop Pro ou mieux encore d'un produit de dessin gratuit comme l'incomparable The Gimp), les systèmes d'exploitation Windows sont accompagnés d'un programme de dessin simple (Paint) qui permet de disposer d'un échantillonneur de couleurs gratuit.

1. Ouvrez Paint (en principe, **Démarrer** > **Tous les programmes** > **Accessoires** > **Paint**).
2. Choisissez **Couleurs** > **Modification des couleurs** . La fenêtre **Modification des couleurs** s'affiche.

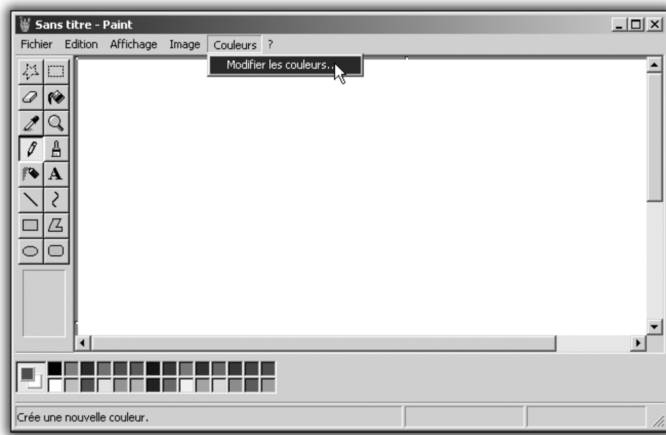


Figure 6.3 : *Le logiciel de dessin Windows Paint*

3 Cliquez sur **Définir les couleurs personnalisées**. La fenêtre **Modification des couleurs** s'agrandit, proposant une zone colorée dans laquelle se trouve un curseur, une zone de visualisation agrandie de la couleur en bas, ainsi que, sur la droite, les valeurs RGB exprimées en nombres décimaux.

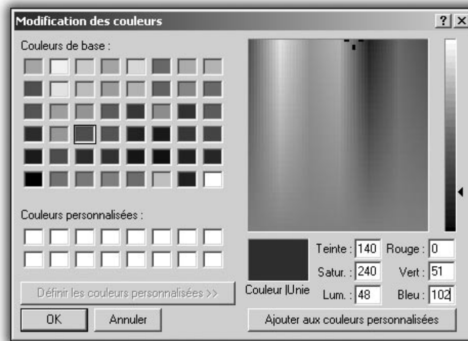


Figure 6.4 : *Fenêtre Modification des couleurs de Paint*

4 Déplacez le curseur et choisissez la couleur souhaitée. Notez par écrit les valeurs RGB, et continuez à rechercher des couleurs ou fermez Paint (cliquez sur **Annuler** puis sur **Fichier** > **Quitter**). C'est tout !

Enfin, pas exactement. Vous disposez de chiffres décimaux qu'il faut convertir en hexadécimal. À moins d'être très doué et de savoir faire cela avec facilité, vous avez besoin d'un convertisseur. La

Calculatrice Windows, également livrée avec le système d'exploitation, est parfaite pour cela.

1. Ouvrez la calculette (**Démarrer** > **Tous les programmes** > **Accessoires** > **Calculatrice**). Passez si nécessaire (c'est probable) en mode scientifique en choisissant **Affichage** > **Scientifique**. Vérifiez que le bouton d'option **Déc.** est sélectionné, juste sous la ligne d'affichage.

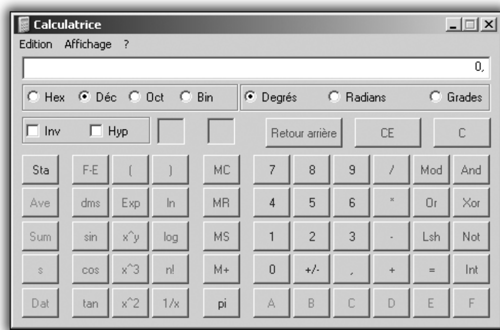


Figure 6.5 : *Calculatrice Windows en mode scientifique*

2. Saisissez le code de la couleur rouge.
3. Cliquez sur le bouton d'option **Hex** et notez par écrit le résultat à deux chiffres (il peut y avoir une lettre) correspondant à la couleur rouge.
4. Répétez ces étapes pour le vert et le bleu, puis assemblez ensuite les différents codes de deux chiffres : RRVVBB. Voilà votre code RGB pour cette couleur.

En vous reportant à la figure montrant la fenêtre **Modification des couleurs** de Paint de la figure 6.4, vous voyez que les valeurs pour la couleur sélectionnée étaient `0`, `51` et `102`, ce qui donne `00`, `33` et `66`, donc un code RGB de `003366` : justement la couleur employée par Micro Application pour sa page d'accueil.



Convertisseur

Remarquez que si vous disposez d'un code RGB en hexadécimal et que vous souhaitez voir à quelle couleur il correspond, accomplissez exactement l'opération inverse : convertissez le code en décimal, puis saisissez directement sous Paint les valeurs obtenues dans les cases adéquates : la couleur correspondante s'ajuste après la modification de chaque valeur, pour aboutir à la couleur cherchée.



On line

Vous pouvez également vous servir d'un convertisseur en ligne, comme celui proposé par AlterLinks à l'adresse **www.alterlinks.fr/conversion-table/RGB-HTML-RWX.php**

Modification de la couleur du texte dans un élément de page

Comme pour l'élément `BASEFONT`, vous pouvez recourir à l'attribut `color` d'un élément `FONT` pour modifier la couleur du texte pour ou dans un élément. Gardez bien présent à l'esprit que toute modification appliquée à l'aide de `FONT` reste active jusqu'à la fermeture de l'élément `FONT` !

Pour renforcer l'impact d'un terme important, vous pouvez combiner l'emploi d'une emphase `STRONG` avec une couleur rouge, par exemple :

```
<P>Pensez &agrave; <STRONG><FONT color="red">sauvegarder
</FONT></STRONG> r&eacute;guli&egrave;rement votre travail !
</P>
```

Le mot « *sauvegarder* » s'affiche en gras et en rouge.



Figure 6.6 : *Modification de la couleur d'un mot dans un élément*

Modification du schéma de couleurs d'un tableau

bgcolor est un autre attribut facultatif de plusieurs éléments de tableau TABLE, TR, TD et TH. De façon analogue à l'attribut bgcolor de l'élément BODY, il modifie la couleur de fond. Il peut donc modifier la couleur de fond de tout le tableau, d'une ligne ou d'une cellule d'en-tête ou de donnée.

Si vous modifiez le fond d'une cellule, assurez-vous bien que cela ne compromette pas la lisibilité du texte qui y est placé : servez-vous de l'élément FONT ou mieux encore d'une feuille de style pour modifier également la couleur du texte. L'attribut bgcolor est désormais déconseillé au profit des feuilles de style. Les fonds colorés de tableaux ne sont pas reconnus par les plus vieux navigateurs.

Modifier la couleur d'une cellule de tableau

Pour modifier la couleur de fond d'une cellule de tableau, insérez l'attribut bgcolor dans l'élément TD, exactement comme vous procéderiez avec un élément BODY ou un élément TABLE.

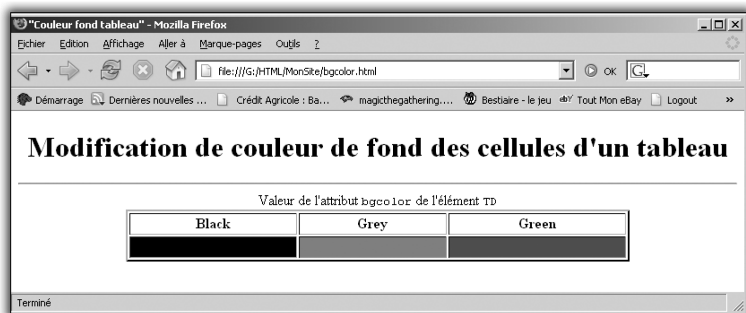


Figure 6.7 : *Modification de la couleur de fond d'une cellule de tableau*

N'oubliez pas de modifier si nécessaire la couleur de la police à l'aide de l'élément FONT !

Précautions à prendre avec les couleurs

L'emploi des couleurs nécessite certaines précautions. Sachez que leur rendu réel dépend de la configuration tant matérielle que logicielle de

l'ordinateur de l'utilisateur. Chaque carte graphique, chaque écran, voire chaque navigateur peut influencer légèrement le rendu. Si la différence est généralement subtile, une configuration inhabituelle peut rendre le site difficile à lire. C'est pourquoi, une fois encore, mieux vaut tester le site sur autant de machines différentes que possible (il suffit d'examiner les rangées de téléviseurs dans n'importe quel magasin d'audiovisuel ou grande surface pour disposer d'une idée des variations susceptibles de se produire selon le matériel employé !). Une bonne astuce consiste souvent à examiner le site à l'aide d'une machine et de logiciels datant un peu, pour voir si le site se « dégrade » de façon acceptable.

Par exemple, vous avez remarqué que le codage RGB permet de disposer de plus de 16 millions de couleurs : si votre écran n'est pas configuré en millions de couleurs, vous ne pourrez voir selon les cas que 65 000 ou même 216 couleurs (utilisation dans ce cas uniquement des nombres `00`, `33`, `66`, `99`, `CC` et `FF`).

Même si les couleurs peuvent apporter beaucoup d'informations aux documents et les rendre plus lisibles, mieux vaut s'en tenir aux principes suivants lors de leur emploi :

- Attention aux combinaisons de couleurs susceptibles de poser problème aux personnes atteintes de daltonisme dans ses diverses formes. Toutes les études démontrent que 10 à 12 % de la population distinguent plus ou moins mal les couleurs. Si cela se limite le plus souvent à une confusion entre le rouge et le vert, certains sont totalement aveugles aux couleurs. Les couleurs possédant la même intensité lumineuse leur sont particulièrement difficiles à distinguer. Lorsque vous définissez les couleurs de fond et de texte, souvenez-vous que des couleurs à fort contraste, noir et blanc ou noir et jaune, sont plus facile à voir et à différencier. Les visiteurs souffrant d'une déficience d'acuité visuelle vous en sauront gré également !
- Si vous employez une image d'arrière-plan ou définissez une couleur d'arrière-plan, veillez à bien assortir les diverses couleurs des textes.
- Les couleurs définies par les éléments `BODY` et `FONT`, et par l'attribut `bgcolor` sur les tables ressortent différemment selon les plates-formes (par exemple, entre les stations de travail et les ordinateurs Macintosh et Windows, et entre les écrans à cristaux liquides et les écrans cathodiques) : vous ne pouvez pas compter uniquement sur un effet spécifique. Dans le futur, la gestion du

modèle de couleur SRGB en même temps que celle des profils de couleur ICC devrait atténuer ce problème.
- La spécification de couleurs dans des éléments HTML est déconseillée. Servez-vous plutôt des feuilles de style.

Enfin, sachez que les couleurs peuvent être désactivées par l'utilisateur. Tous les navigateurs ou presque possèdent désormais des options de gestion des couleurs ou autorisent les utilisateurs à définir un ensemble particulier de couleurs à utiliser pour tout site visité. Assurez-vous que vos graphismes ne risquent pas de disparaître ou de perdre toute signification si un autre schéma de couleur est retenu. Les schémas les plus classiques sont les suivants :

- Fond blanc, texte noir, liens hypertextes bleus.
- Fond noir, texte blanc et liens hypertextes jaunes.
- Fond gris, texte noir et liens hypertextes bleus.

Qu'il s'agisse là des schémas les plus fréquents ne signifie pas qu'ils soient les seuls. Certains utilisateurs peuvent avoir retenu des couleurs à contraste maximal, d'autres peuvent avoir augmenté fortement la taille des polices afin de lire plus aisément les textes.

6.2. Les images

Vous le savez déjà, il est possible de placer des images sur un site Web. Les pages Web ont le plus souvent recours à deux formats de fichiers graphiques, GIF (*Graphic Interchange Format*) et JPEG (*Join Photographic Experts Group*), même s'il est parfois possible de rencontrer des images PNG (*Portable Network Graphic*). Le format JPEG, qui accepte 16,7 millions de couleurs, est surtout employé pour des photographies et les graphismes dotés de nombreuses couleurs. Leur système de compression leur permet de diminuer la taille totale des fichiers. En revanche, le format GIF, limité à une palette maximale de 256 couleurs, sert plutôt pour les dessins au trait, les icônes et les images animées.

Le format PNG, libre de droits, est annoncé comme le successeur du format propriétaire GIF. PNG prend en charge trois types d'images : vraies couleurs, échelle de gris et fondée sur une palette (« 8 bits »), tandis que JPEG ne prend en charge que les deux premiers et GIF uniquement le troisième, même s'il peut émuler une échelle de gris à

l'aide d'une palette de gris. PNG gère en outre la transparence d'une façon plus sophistiquée que GIF. Il s'agit d'un format destiné aux images fixes, même si un projet nommé MNG pour les images animées a vu le jour en 1999. La compression PNG est d'excellente qualité.

En pratique, le choix du format se limite souvent à la version de l'image qui occupe le moins de place. Le temps de chargement d'une page étant proportionnel au nombre d'éléments graphiques qu'elle contient ainsi qu'à leur taille totale, il est capital de retenir des images de la plus faible taille possible. Il est prouvé qu'une page demandant plus de 15 secondes de téléchargement a tendance à chasser les visiteurs potentiels (attention : en situation réelle, pas en local sur votre ordinateur !).

Quel que soit le format d'image retenu, l'inclusion dans une page Web reste identique.

Image d'arrière-plan

Au lieu d'un fond de couleur uniforme, vous pouvez utiliser une image d'arrière-plan *(background)*, spécifiée dans l'élément `BODY` à l'aide de l'attribut `background`. La syntaxe est la suivante :

```
<BODY background ="nomfichier">
```

où « *nomfichier* » est le nom du fichier image à utiliser comme fond d'écran. Cet attribut est désormais déconseillé, au profit des feuilles de style.

Une telle image remplit l'écran par un effet de mosaïque. Autrement dit, elle est répétée en largeur autant de fois que nécessaire pour occuper la largeur de l'écran et est de même répétée en hauteur autant de fois que nécessaire pour remplir toute la hauteur de l'écran.

Évitez les très grands graphismes en fond d'écran : cela fonctionne rarement bien, et les utilisateurs disposant de grands écrans voient aisément la répétition du graphisme.

Une image d'arrière-plan est prioritaire sur une couleur d'arrière-plan avec la plupart des navigateurs.

Ce type d'image est principalement employé de deux façons : pour créer un effet de bordure (à droite ou à gauche) ou comme mosaïque filigranée sur tout l'écran.

La première solution a recours à une image de hauteur relativement faible (par exemple, une spirale, ou des petits carreaux avec une marge) et de largeur égale à la largeur conseillée pour le site concerné. Ainsi, la répétition est en théorie uniquement verticale, et peut créer un effet de cahier à spirales ou de cahier d'écolier à carreaux, avec un trait de marge en couleur. Inconvénient évident, si la largeur d'écran employée n'est pas exactement celle prévue, la répétition peut également se produire horizontalement, détruisant toute cette belle présentation.

Vous pourrez également parfois rencontrer une image ne mesurant que quelques pixels de haut et de grande largeur. L'auteur a pris la précaution de s'assurer d'une apparente continuité entre le haut et le bas de son image, ainsi qu'entre les côtés droit et gauche : toute répétition semble alors presque invisible.



Figure 6.8 : *Une image ayant la forme d'un fin bandeau.*



Figure 6.9 : *Un fin bandeau employé comme image de fond est une solution séduisante de prime abord…*

L'image employée ici, qui mesure 802 pixels de large sur 32 de haut, est adaptée à une fenêtre de navigateur d'environ 800 pixels de large (soit une résolution de 800 x 600, pour un affichage en plein écran). Si la fenêtre est plus petite ou plus grande, tout l'aspect est compromis.



Figure 6.10 : *Un aspect modifié par l'utilisateur*



Figure 6.11 : *… mais qui reste largement dépendante de l'utilisateur !*

La seconde solution a recours à une image de fond de relativement petite taille (plus la taille est petite et plus les erreurs de répétition sont imperceptibles), très peu colorée et contrastée, servant ainsi de « filigrane » au reste du contenu. J'avoue affectionner particulièrement ce type de fond, que je trouve efficace, sobre et élégant.



Figure 6.12 : *L'emploi d'un filigrane.*

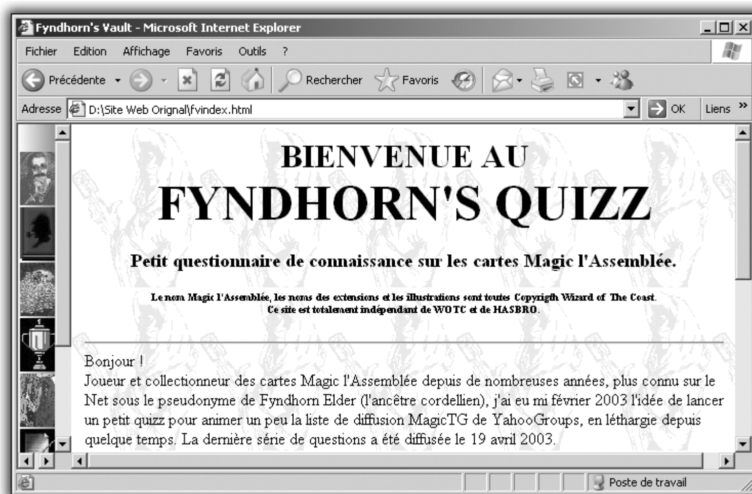


Figure 6.13 : *L'emploi d'un filigrane.*



Figure 6.14 : *L'emploi d'un filigrane de faible taille (ici 129 × 159 pixels) est une solution possible, indépendante de la taille de l'écran employé.*

Image insérée

Vous pouvez également insérer des images pratiquement dans tout élément de votre page : une image peut donc se trouver en plein milieu d'un texte, ou ce dernier s'enrouler en colonnes autour du dessin.

Avant d'envisager de diffuser une image sur Internet, vous devez prendre certaines précautions (comme d'ailleurs avec pratiquement tout contenu) : veillez à toujours respecter scrupuleusement la législation sur le copyright et les droits d'auteur. Pour pouvoir diffuser une image sur votre site, vous devez vérifier que vous avez le droit de le faire : soit elle est libre de droits, soit vous avez obtenu l'accord de son propriétaire

(auteur), soit enfin vous en êtes vous-même l'auteur. Même dans ce dernier cas, si l'image est une photographie représentant d'autres personnes, vous devez obtenir leur autorisation formelle.

Depuis bien longtemps, HTML permet l'inclusion d'images à l'aide de l'élément `IMG`. Cet élément incorpore une image dans le document courant, à l'emplacement de la définition de l'élément. Dépourvu de contenu, il est généralement remplacé dans la ligne par l'image que désigne l'attribut `src`, les images alignées à gauche ou à droite qui « flottent » hors de la ligne faisant exception.

Nous allons modifier l'apparence de la page secondaire *Région* en insérant une image. Pour simplifier, nous allons employer une image dont disposent normalement tous les utilisateurs de machines Windows XP : une de celles se trouvant dans le dossier *Echantillons d'images* du dossier *Mes images*. Avant de commencer, il faut vous livrer à quelques travaux préparatoires :

1 Ouvrez l'Explorateur Windows, naviguez jusqu'à votre dossier *MonSite* et créez-y un sous-dossier, nommé *Images* : vous y placerez toutes les images employées sur le site. C'est une bonne habitude à prendre pour éviter de surcharger le répertoire principal.



Subdivisions

Vous pourriez même aller encore plus loin et subdiviser le sous-dossier *Images* en plusieurs sous-dossiers : *gif* (les images GIF statiques), *gifanim* (les gif animés), *icone* (les icônes et boutons) et *JPEG* (les photos).

2 Choisissez **Démarrer** > **Mes images**, puis cliquez sur le dossier *Echantillons d'images*.

3 Sélectionnez l'image *Coucher de soleil.jpg*, puis choisissez dans la barre de menu **Edition** > **Copier**.

4 Revenez dans votre dossier *MonSite/Images*, puis collez l'image de coucher de soleil en choisissant **Edition** > **Coller**.

5 Procédez de la même façon pour les images *Hiver.jpg* et *Nénuphars.jpg*.

Vous êtes maintenant prêt à insérer des images sur votre site.

6 Ouvrez le Bloc-Notes, puis ouvrez le fichier *region1_4_3.html*.

7 Modifiez d'abord le numéro de version :

```
<META name ="version" content="1.6.1">
```

8 Profitez-en pour modifier le lien vers la page d'accueil en fonction du nom du fichier actuel :

```
<A href="pageacc1_6_1.html">Retour vers la page d'accueil
</A>
```

9 Placez-vous maintenant dans la première cellule de la première ligne, et remplacez le contenu texte « `...Une image sera placée ici...` » par un élément `IMG`, comme suit :

```
<TD rowspan="3"><IMG src="Images/Coucher de soleil.jpg">
</TD>
```

Remarquez que la balise `IMG` est dépourvue d'élément de fermeture : son contenu est défini par la valeur de l'attribut `src`, un URI. Cet attribut indique au navigateur où trouver l'image, comme l'attribut `href` de l'élément `A` indique où trouver la ressource liée à une ancre.



Sensibilité à la casse

Respectez avec soin la casse dans l'URI de l'élément `IMG`, comme pour tout URI. Certains navigateurs et/ou serveurs sont sensibles à la casse des URI. Tous ne l'étant pas, votre site pourrait fonctionner parfaitement en local, mais plus du tout une fois en ligne. Une bonne habitude consiste à n'employer que des minuscules pour les noms des fichiers image.

Cette sensibilité à la casse étant susceptible de concerner également l'extension, vérifiez celles de vos fichiers image : certains programmes de dessin enregistrent automatiquement les fichiers avec une extension en majuscules.



XHTML

En XHTML, cette instruction s'écrirait :

```
<img src="Images/Coucher de soleil.jpg" />
```

10 Enregistrez le fichier sous le nom *region1_6_1.html*, puis examinez cette page dans votre navigateur.



Archivez…

L'habitude prise ici d'attribuer aux fichiers un nom comportant le numéro de version n'a pour but que d'éviter les recouvrements de fichiers et de permettre d'étudier les versions précédentes. Pour un site réel, cette méthode n'est guère à recommander, la modification des noms de fichiers entraînant des risques de liens rompus et imposant de mettre continuellement à jour ces liens. Mieux vaut archiver les anciens fichiers, puis les modifier sans changer leur nom.



Figure 6.15 : Insertion d'une image

L'aspect n'est pas extraordinaire, et ce pour plusieurs raisons. Nous ne commenterons pas le choix de l'image, puisqu'il a fallu employer une image préexistante sur la majorité des systèmes : le choix est très limité. En revanche, cette image semble de taille un peu exagérée, « écrasant » tout le tableau et nécessitant un écran de grande taille. En examinant le fichier image, vous voyez que celle-ci mesure 800 × 600 pixels : une taille imposante, sans doute démesurée par rapport à nos besoins. Il

faudrait l'afficher en taille plus réduite. C'est ce que permettent les attributs `height` et `width` de l'élément `IMG`.

11 Modifiez comme suit l'élément `IMG` en ajoutant les attributs `height` et `width` :

```
<TD rowspan="3">
<IMG src="Images/coucher de soleil.jpg " width="267"
     height="200">
</TD>
```

12 Enregistrez à nouveau le fichier sous le même nom puis examinez la page dans votre navigateur.

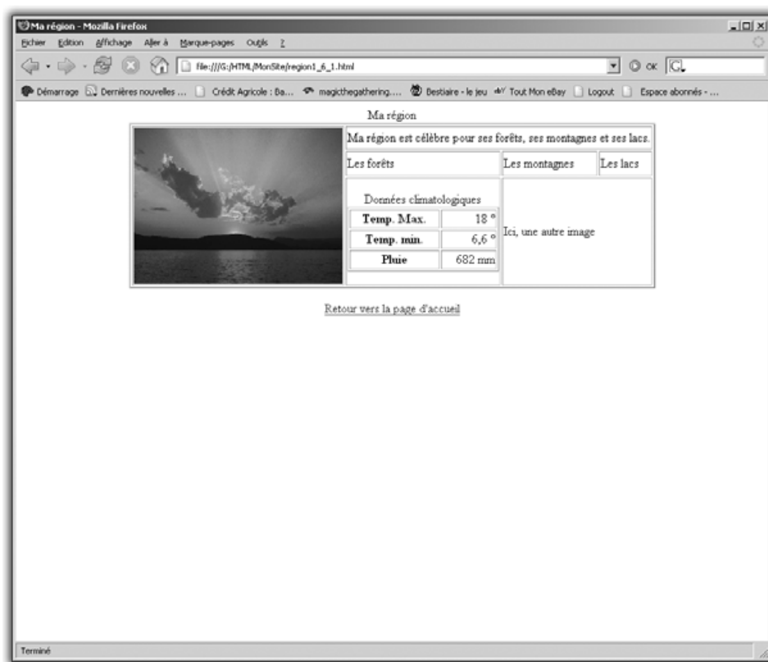


Figure 6.16 : *Insertion d'une image à taille déterminée*

Nous avons ici divisé la taille d'affichage de l'image par 3. L'aspect est nettement plus satisfaisant (remarquez que nous avons conservé exactement la même taille de fenêtre que pour la capture d'écran précédente).

Ces deux attributs méritent quelques commentaires supplémentaires :

- Leurs valeurs sont exprimées en pixels.

- Les deux attributs ne sont pas obligatoires si vous souhaitez modifier la taille d'une image de façon proportionnelle : ne saisir que `height="200"` suffit à réduire la taille de l'image d'un tiers en hauteur et en largeur. La présence des deux attributs permet en outre de modifier le ratio de l'image, ce qui est parfois utile.
- Ces attributs ont été employés ici pour réduire la taille de l'image, mais ils auraient tout aussi bien pu servir à l'augmenter. Selon le rapport d'agrandissement retenu, attendez-vous à une perte de qualité (pixellisation de l'image).
- Lorsque vous incluez ces attributs, la page Web semble se charger plus vite : le navigateur est en mesure d'afficher le texte avant d'avoir fini de charger la page, car il connaît la place à réserver à l'image. En leur absence, le visiteur est confronté à une page blanche pendant le chargement de l'image, le navigateur ne sachant pas la place libre dont il dispose pour afficher le texte. Ce phénomène est presque imperceptible en local, mais devient évident lorsque la page est téléchargée depuis le Web.

En pratique, vous chargez ici une image d'un « poids » environ neuf fois supérieur à ce qui est nécessaire : 800 × 600 pixels, soit 480 000 pixels, alors que vous n'exploitez qu'une image de 266 × 200 pixels, soit 53 200 pixels. Malgré la compression JPEG, le temps de chargement est environ neuf fois supérieur aux besoins. Mieux vaut toujours redimensionner les images à la taille souhaitée à l'aide d'un programme de dessin (nous reviendrons par la suite sur ce thème). En revanche, conservez donc les attributs `height` et `width` pour accélérer l'affichage de la page.

13 Lorsque vous insérez une image, vous devez penser aux agents utilisateurs non visuels. Servez-vous de l'attribut `alt` pour proposer une alternative textuelle dès le début du chargement de l'image :

```
<IMG src="Images/coucher de soleil.jpg " width="267"
     height="200"
     alt="Coucher de soleil sur le lac de Linciel">
```

14 Enregistrez à nouveau le fichier sous le même nom puis examinez la page dans votre navigateur.

L'attribut `alt` spécifie le texte de remplacement qui est restitué si l'image ne peut s'afficher. Les agents utilisateurs doivent restituer le texte de remplacement quand ils ne gèrent pas les images, ne

reconnaissent pas un certain type d'image ou ne sont pas configurés pour les afficher.

Remarquez qu'Internet Explorer affiche le texte alternatif dans une infobulle lorsque le pointeur de la souris est placé sur l'image.



Figure 6.17 : *Image avec texte alternatif*

C'est une des nombreuses erreurs liées au manque de respect d'Internet Explorer vis-à-vis de la spécification HTML 4.01 : la valeur de l'attribut `alt` ne devrait être affichée que lorsque l'agent utilisateur ne peut afficher l'image. Pour afficher un commentaire dans une infobulle, vous devez employer l'attribut `title`.

15 Servez-vous de l'attribut `title` pour proposer un commentaire affiché dans une infobulle lorsque le pointeur se trouve sur l'image :

```
<IMG src="Images/coucher de soleil.jpg "
     width="267" height="200"
     alt="Coucher de soleil sur le lac de Linciel"
     title="Un coucher de soleil sur le lac de Linciel">
```

16 Enregistrez à nouveau le fichier sous le même nom puis examinez la page dans votre navigateur.

Remarquez que nous avons volontairement différencié les valeurs des attributs `alt` et `title`. Tant Internet Explorer que Firefox affichent désormais le commentaire dans une infobulle. Internet Explorer

n'affiche le contenu d'un éventuel attribut `alt` qu'en l'absence de l'attribut `title`.



Figure 6.18 : Emploi de l'attribut title.



Figure 6.19 : Emploi de l'attribut title. Tant Internet Explorer que Firefox affichent correctement ce commentaire dans une infobulle lorsque le pointeur se trouve sur l'image.

Vous pouvez également recourir à l'attribut `longdesc` pour relier l'image à une description plus détaillée. Cet attribut se comporte comme une ancre :

```
<IMG src="image.jpgf" alt="Schéma du moteur "
     longdesc="schémamoteur.html">
```

L'élément `IMG` possède d'autres attributs qui permettent de jouer sur les paramètres de restitution d'une image :

- L'attribut `align` permet d'aligner l'image, de la même façon que pour un paragraphe. Vous pouvez utiliser les valeurs `center`, `right` ou `left`. Si vous placez du texte après le graphisme, celui-ci coulera autour de l'image alignée à droite ou à gauche, donnant un aspect similaire à celui d'un magazine.
- L'attribut `border` permet de définir la largeur du cadre entourant l'image. En choisissant `border="0"`, l'image sera dépourvue de cadre.

6.3. Images animées

Pour ajouter de l'éclat à une page, il est possible d'utiliser des gifs animés. Il s'agit d'un type d'image GIF qui peut être animé en combinant plusieurs images en un unique fichier GIF, selon le standard défini en 1987 par Compuserve. L'image effectue lors de sa restitution un cycle à travers les différentes images élémentaires.

Même si les animations GIF n'offrent pas le niveau de souplesse et de contrôle proposé par d'autres formats d'animation, il a connu et connaît encore un immense succès largement dû au fait qu'il est reconnu par l'immense majorité des navigateurs. En outre, les fichiers de gifs animés sont généralement plus petits que d'autres fichiers d'animation, comme les applets Java. Ce type d'image est souvent employé pour les bannières publicitaires, pour attirer des visiteurs sur une page, mais peut également servir à attirer l'attention sur des parties précises d'une page ou comme boutons de barre de navigation.

GIF89a, comme GIF, possède une palette de couleurs comprise entre 2 et 256 couleurs. Plus la palette est petite et plus le fichier est de petite taille. Si votre image GIF n'emploie que 4 couleurs, vous pouvez réduire en conséquence la palette à 4 couleurs (2 bits) et ainsi diminuer la taille du fichier de près de 75 %.

Insertion d'un gif animé

L'insertion d'un gif animé est strictement identique à celle d'une image statique, comme vu précédemment.

Vous allez maintenant ajouter un gif animé à votre page d'accueil, pour mieux attirer l'attention sur la ligne permettant de vous envoyer un courriel. Problème : vous ne disposez pas d'un tel gif animé. Qu'à cela ne tienne, vous pouvez heureusement trouver sur le Net de nombreuses images libres de droits, que vous pouvez employer en toute liberté pour vos créations. Ouvrez n'importe quel outil de recherche, tapez comme mots-clés `GIF` et `gratuit` et vous n'avez que l'embarras du choix.



Figure 6.20 :
Une recherche simple procure des milliers d'adresses potentielles

1 Ouvrez votre navigateur, puis connectez-vous à Internet (si ce n'est déjà fait). Entrez l'adresse suivante dans la barre d'adresse de votre navigateur : `http://gif.webgratuit.com/`

2 Il s'agit d'une des adresses, choisie presque au hasard, de la recherche précédente. Il en existe bien d'autres… Dans la fenêtre qui s'affiche, cliquez dans le cadre du milieu sur **Gif/Gif animés**.



Figure 6.21 :
Page d'accueil de WebGratuit

Dans la liste qui s'affiche, choisissez **Webdesign** (en bas de la colonne de gauche), puis **Mail** (dans la colonne de droite), puis enfin **Boîtes**. La page qui apparaît propose différentes images animées de boîtes aux lettres.



Figure 6.22 : *Gifs animés de boîtes aux lettres*

3 Descendez si nécessaire dans la fenêtre pour vous placer sur la seconde boîte aux lettres rouge sur fond noir (nommée *boites009.gif*). Appuyez sur le bouton droit de votre souris et choisissez **Enregistrer la cible sous...** .

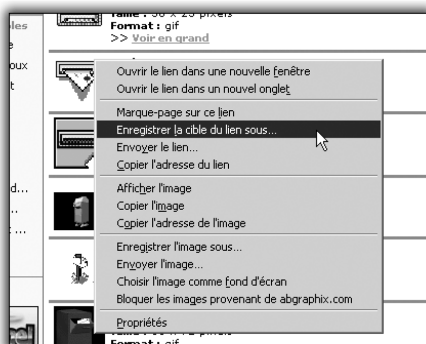


Figure 6.23 : *Enregistrement d'un gif animé*

4 Dans la boîte de dialogue **Enregistrer sous**, sans modifier le nom de fichier, choisissez le dossier de destination en naviguant jusqu'à *MonSite/Images*, puis cliquez sur **Enregistrer**.

5 Dans le Bloc-Notes, ouvrez le fichier *pageacc1_6_1.html*. Modifiez comme suit la ligne d'envoi de courriel :

```
<A href="mailto:votre_nom@votre_FAI">
  <IMG src="Images/Boites009.gif">Ecrivez-moi !
</A>
```

6 Enregistrez le fichier sans modifier son nom, puis examinez-le dans le navigateur.



Figure 6.24 : *Page d'accueil modifiée*

Vous comprenez pourquoi nous avons choisi un gif à fond noir : pour que l'image se mêle agréablement au fond, défini comme noir (`bgcolor="black"`). En revanche, l'aspect n'est pas très agréable : le cadre jaune choque, tandis que la phrase « Écrivez-moi ! » fait double emploi.

7 Rectifiez comme suit l'élément `IMG` :

```
<A href="mailto:votre_nom@votre_FAI">
  <IMG src="Images/Boites009.gif" border="0"
       alt="Ecrivez-moi !">
</A>
```

8 Enregistrez le fichier, puis rafraîchissez votre navigateur.



Figure 6.25 : *Page d'accueil modifiée*

L'aspect est plus agréable. Vous venez de comprendre en quoi l'harmonisation des couleurs d'une page et de celles des éléments graphiques qui y sont insérés est capitale.

Prenez garde à toujours sélectionner des couleurs en harmonie avec les images et les icônes de votre page. Une fois les couleurs choisies, vous pouvez les appliquer à votre fond, au texte et aux liens hypertextes selon vos désirs. Inversement, si vous avez déjà sélectionné les couleurs (ou qu'une charte graphique est imposée), soyez prudent dans le choix de vos images.



Veillez au fond

Les images GIF peuvent posséder une couleur transparente (généralement celle d'arrière-plan), ce qui facilite grandement leur insertion sur n'importe quel fond. Tout arrière-plan coloré, ou fondé sur une image, apparaîtra derrière le nouveau dessin. Veillez toutefois à ce que l'image contraste agréablement avec le fond retenu.



La sélection ou la modification de la couleur transparente s'effectue à l'aide de votre logiciel de dessin favori : reportez-vous à sa documentation.

Création d'un gif animé

S'il existe sur le Web autant de gifs animés proposés en téléchargement gratuit, cela n'est pas un hasard : la création de gifs animés est d'une grande facilité, ce qui en a fait un sport international très prisé. Notez qu'une image animée est chargée une seule fois, quel que soit le nombre de répétitions programmées.

Il a été dit plus haut qu'il s'agissait d'une succession d'images GIF assemblées en un unique fichier GIF : voyez-la un peu comme un dessin animé, où chaque image présente une légère différence avec la précédente. En partant d'une première image fixe,

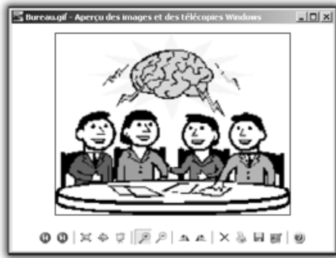


Figure 6.26 :
Image fixe initiale

de taille relativement simple, il est assez facile de la dupliquer, de la modifier légèrement,



Figure 6.27 :
Image modifiée : la couleur de fond a été changée et définie comme transparente, et seul reste un petit « éclair » partant du cerveau.

de l'enregistrer sous un nouveau nom, et ainsi de suite, jusqu'à disposer d'une série d'images élémentaires.



Figure 6.28 :
Les différentes images élémentaires, chacune avec un éclair (total ou fractionnel) différent. Il faut ensuite les assembler.

Pour ce faire, vous avez à votre disposition un certain nombre d'outils, qui relèvent comme à l'habitude des produits commerciaux ou des logiciels gratuits. Vous trouverez les adresses menant vers plusieurs d'entre eux dans l'Annexe E, consacrée aux ressources Web. J'éprouve une légère préférence (irrationnelle ?) pour deux d'entre eux, probablement due au fait que je les emploie depuis longtemps. Le premier, Microsoft Gif Animator version 1.0.0.101, assez ancien (il a été conçu pour Windows 95/98), est quelque peu spartiate. Sa simplicité le rend toutefois particulièrement simple à employer. L'autre, Ulead Gif Animator 1.5, également ancien (1997) est plus complexe. Il dispose en particulier d'options intéressantes permettant de diminuer la taille du fichier résultant… Il est fréquent que je me serve des deux en combinaison, l'un après l'autre : vous comprendrez pourquoi un peu plus loin.

Si Ulead Gif Animator 1.5 était à l'époque disponible gratuitement, les versions ultérieures sont payantes. Une version de démonstration de Ulead Gif Animator 5 (49,95 $ pour la version complète), peut être téléchargée à l'adresse **www.ulead.com**. Heureusement, Microsoft Gif Animator reste disponible gratuitement, notamment à l'adresse **www.zdnet.fr/telecharger/windows/fiche/0,39021313,11010272s,00.htm**

Une fois Microsoft Gif Animator téléchargé et installé, vous y importez successivement les différentes images élémentaires.

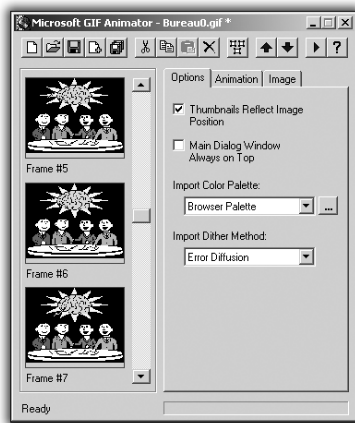


Figure 6.29 :
Microsoft Gif Animator avec les images élémentaires chargées

Là commence le vrai travail : vous pouvez jouer sur les différents paramètres. L'onglet **Options** doit rester en principe inchangé. L'onglet **Animation** également, sauf en ce qui concerne le réglage *Lopping* : celui-ci détermine si l'animation doit se répéter et, le cas échéant, un certain nombre de fois ou indéfiniment. Si l'option *Loop forever* est la plus employée, dans certains cas une seule diffusion de l'animation est préférable.

Sur l'onglet **Images**, le réglage *Duration* est le plus important : il définit le temps, exprimé en centièmes de secondes, pendant lequel est affichée l'image élémentaire active. En cliquant dans la barre d'outils sur **Select All**, vous pouvez appliquer un paramètre à l'ensemble des images, puis l'ajuster par la suite pour chaque image.

Il est possible de modifier l'ordre des images mais aussi d'ajouter (ou de répéter) ou supprimer des images élémentaires. Dans tous les cas, un clic sur le bouton **Preview** permet d'obtenir un aperçu de l'image.



Figure 6.30 :
Aperçu de l'animation dans Microsoft Gif Animator

Une fois satisfait du résultat, il ne vous reste qu'à enregistrer le fichier de gif animé sous le nom et à l'emplacement de votre choix.

Nous avons cependant signalé qu'il était important de chercher à optimiser en la diminuant la taille de tout fichier téléchargé pour une page Web. C'est là qu'intervient pour moi Ulead Gif Animator 1.5,

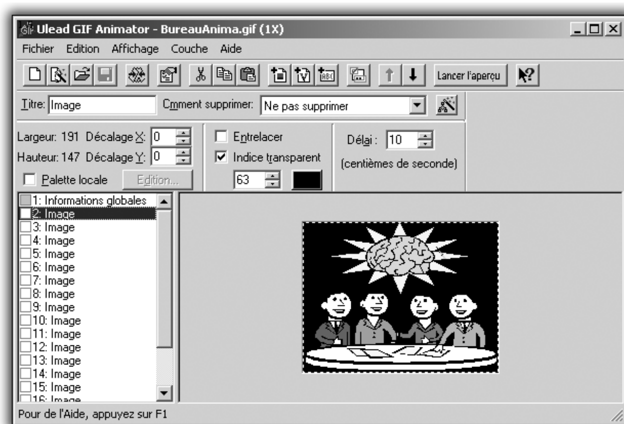


Figure 6.31 : *Ulead Gif Animator*

dont l'utilitaire d'optimisation des gifs animés est remarquable.



Figure 6.32 : *Un autre écran de l'Assistant Optimisation d'Ulead Gif Animator*



Figure 6.33 :
encore un autre…



Figure 6.34 :
La diversité des écrans de l'Assistant Optimisation d'Ulead Gif Animator



Figure 6.35 :
Différents écrans de l'Assistant Optimisation d'Ulead Gif Animator

Remarquez que la taille du fichier a été très fortement diminuée : passer de presque deux dixièmes de seconde à un centième de seconde de temps de téléchargement n'offre rien d'anodin. Soyons tout de même honnête : cela est estimé à partir des temps de chargement de l'époque (soit à l'aide d'un modem à 28,8 kbps). Si le rapport de grandeur reste le même aujourd'hui, avec une connexion à 500 kbps (presque le minimum d'une connexion câble ou ADSL), les temps de chargement sont respectivement 1 centième de seconde et 0,5 millième de seconde.

Même si la page comporte cent images animées, la différence est presque imperceptible pour l'utilisateur. Il demeure toutefois que, par respect pour les utilisateurs disposant de connexions plus lentes tant que pour préserver la bande passante globale, mieux vaut toujours réduire la taille des fichiers téléchargés.

Il existe d'autres produits similaires, dont A Smaller Gif, disponible à l'adresse **www.peda.com/smaller/download.html**

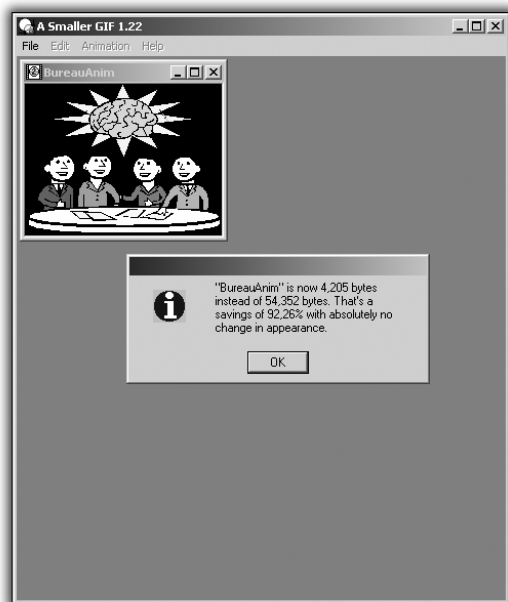


Figure 6.36 :
Optimisation à l'aide de A Smaller Gif

Une recherche devrait également vous fournir des adresses d'outils en ligne.

Parmi les nombreux autres célèbres utilitaires de création de gifs animés, il convient également de citer Animated Shop 3 (29 $, mais livré gratuitement en accompagnement de l'excellent Paint Shop Pro 9, 129 $: **www.jasc.com/products/paintshoppro/**) et Gif gIf giF (**www.peda.com /ggg/download.html**). N'hésitez pas à télécharger les versions de démonstration pour vous familiariser avec ces produits et éventuellement compléter ou remplacer le produit gratuit de Microsoft.



Emploi d'icônes de puces pour une liste désordonnée

Vous avez certainement déjà remarqué de splendides listes, avec des puces colorées figurant en tête de chaque membre de la liste. Il est facile d'en faire autant, même si cela diverge quelque peu du standard HTML. Le recours à une icône de puce « rompt » la structure de la liste, certains agents utilisateurs risquant alors de ne pas la reconnaître comme telle. Au lieu d'employer une liste (élément `UL`), placez simplement un élément saut de ligne (`BR`) après chaque membre de la liste. Au début de la ligne suivante, insérez votre icône de puce (statique ou animée), par exemple comme suit :

```
<IMG src="images/puce.gif" height=5 width=5 alt="*">
```

Avec la majorité des navigateurs, vous ne verrez aucune différence visuelle avec une « vraie » liste (voir les figures suivantes, présentant respectivement une telle liste et le code source correspondant). Même si cela est visuellement attrayant, gardez en mémoire qu'il s'agit d'une atteinte à la structure du document…



Figure 6.37 : *Exemple de liste recourant à des icônes de puces*

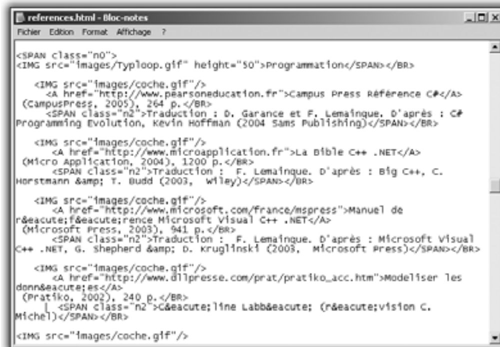


Figure 6.38 : *Fragment du code source de la page précédente*

6.4. Image cliquable

Une image cliquable permet de spécifier des régions (ou un objet) dans l'image et d'affecter une action particulière à chacune de ces régions (par exemple, ouvrir une nouvelle page, lancer un programme, etc.). Un clic sur la région (une activation) exécute l'action. Pour créer une image cliquable, vous associez à un objet (ici à une image) une spécification de zones géométriques sensibles sur celui-ci.

Les images cliquables sont de deux types : côté client et côté serveur.

- *Côté client* : elles sont entièrement gérées par l'agent utilisateur de l'utilisateur. En raison de l'absence d'appel au serveur pour l'exécution côté client, elles s'exécutent plus rapidement. Il est de plus possible de les tester en local sur son ordinateur personnel. Lorsque l'utilisateur active une région d'une image cliquable côté client avec une souris, l'agent utilisateur interprète les coordonnées du pixel. L'agent utilisateur sélectionne le lien spécifié pour la région activée et le suit.
- *Côté serveur* : quand l'utilisateur active une région d'une image cliquable côté serveur, les coordonnées du pixel du clic sont envoyées à l'agent côté serveur, spécifié par l'attribut `href` de l'élément `A`. L'agent côté serveur interprète ces coordonnées et effectue une certaine action.

Les images cliquables côté client sont préférées à celles côté serveur pour au moins deux raisons : elles sont accessibles aux personnes navigant avec des agents utilisateurs non graphiques et elles renvoient une réponse immédiate quant à la présence du pointeur sur une région active.

Nous allons examiner tour à tour plus en détail ces deux types.

Images cliquables côté client : éléments MAP et AREA

Les images cliquables côté client ont recours à deux éléments, `MAP` et `AREA`.

Une image cliquable côté client est spécifiée à l'aide de l'élément `MAP`. Elle est associée à un autre élément (ici `IMG`, mais ce pourrait être d'autres objets, comme nous le verrons plus tard) *via* l'attribut `usemap` de celui-ci.

Pour voir comment mettre en œuvre une image cliquable, nous allons d'abord rassembler le matériel nécessaire : en l'occurrence, une simple image de fond de carte, récupérée sur l'excellent site de Sciences-Po.

1 Ouvrez votre navigateur, puis naviguez jusqu'à l'URI *www.sciences-po.fr/cartographie/* .



Figure 6.39 : *Site du département Cartographie de Sciences-Po*

2 Sélectionnez **Fonds de cartes**, puis dans la page qui s'affiche descendez jusqu'à la section **Etats** et sélectionnez la carte *France* (régions).

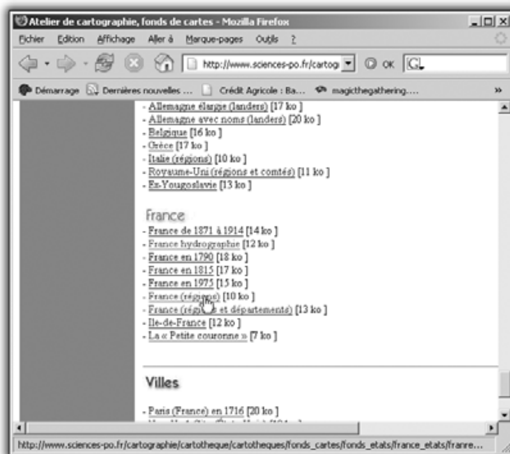


Figure 6.40 : *Page Fonds de cartes*

Effectuez sur ce fichier un clic avec le bouton droit de la souris et choisissez **Enregistrer sous**. Comme vous l'aviez fait précédemment, naviguez jusqu'à votre dossier *MonSite/Images* pour y enregistrer le fichier (nommé *franregs.gif*).

C'est fait : vous disposez d'une image prête à être transformée en image cliquable. Poursuivons en créant la page Web qui va l'employer.

3 Ouvrez le Bloc-Notes, puis ouvrez le fichier *region1_6_1.html*. Il va nous servir de squelette pour créer une nouvelle page.

4 Modifiez d'abord le contenu de la balise `TITLE` :

```
<TITLE>R&eacute;gions fran&ccedil;aises">
```

5 Supprimez tout le contenu de l'élément `BODY`.

6 Entrez dans l'élément `BODY` un nouveau titre de niveau 1 :

```
<H1 align="center">Carte de France avec
r&eacute;gions</H1>
```

7 Placez-vous en dessous et créez comme suit un élément `IMG` inclus dans un élément `P` :

```
<P><IMG src="Images/franregs.gif" usemap="#freg"
alt="Carte de France">
</P>
```

Remarquez l'ajout dans la balise de l'élément `IMG` d'un nouvel attribut, `usemap`. Celui-ci signale que l'image spécifiée par l'URI valeur de l'attribut `src` va être une image cliquable côté client, définie par un élément `MAP`. La valeur de l'attribut `usemap` de l'élément `IMG` doit correspondre avec celle de l'attribut `name` de l'élément `MAP` associé.

8 Ajoutez ensuite dans l'élément `P`, après l'élément `IMG`, un élément `MAP` associé pour le moment vide :

```
<MAP name="freg">
</MAP>
```

La valeur de l'attribut `name` de l'élément `MAP` correspond bien à celle de l'attribut `usemap` de l'élément `IMG`.

9 Par sécurité, enregistrez votre fichier sous le nom *FranceReg.html*.

Le second type d'élément capital pour une image cliquable est l'élément `AREA`. Cet élément possède deux attributs : l'attribut `shape`, qui spécifie la forme d'une région, et l'attribut `coords`, qui spécifie la position et la forme de la région à l'écran.

10 Ajoutez ce qui suit dans l'élément `MAP` :

```
<MAP name="freg">
<AREA shape="rect" coords="161,263,204,323"
      href="Poitou-Charentes.html"
      alt = "R&eacute;gion Poitou-Charentes"
      title="Poitou-Charentes">
<AREA shape="circle" coords="274,167,21"
      href="Ile de France.html"
      alt="R&eacute;gion Ile de France"
      title="Ile de France">
<AREA shape="polygon"
      coords="243,172,286,209,271,283,225,283,197,235,243,172"
      href="Centre.html"
      alt="R&eacute;gion Centre"
      title="Centre">
</MAP>
```

11 Enregistrez à nouveau votre fichier, puis examinez-le dans votre navigateur.

Remarquez que, lorsque vous déplacez le pointeur sur une des zones définies, celui-ci se transforme en main, indiquant la présence d'un lien. En outre, si vous le laissez un peu plus longtemps sur la zone, la valeur de l'attribut `title` s'affiche.



Figure 6.41 :
Aspect de l'image cliquable

Si vous cliquez sur une des zones, vous obtenez un message d'erreur : les pages secondaires spécifiées par les valeurs des attributs `href` n'existent pas !

L'attribut `shape` accepte quatre valeurs possibles, dont trois sont employées dans l'exemple précédent :

- `default` spécifie la région entière.
- `rect` définit une région rectangulaire.
- `circle` définit une région circulaire.
- `poly` définit une région polygonale.

Le nombre et l'ordre des valeurs de l'attribut `coords` dépendent de la valeur de l'attribut `shape`, comme le montre le tableau suivant. Il s'agit d'une liste de valeurs longueur, séparées par une virgule.

Tableau 6.2 : *Valeurs des attributs shape et coords d'un élément AREA*

Valeur de l'attribut `shape`	**Formes**	**Coordonnées (**`coords`**) nécessaires**
`default`	Toute l'image	Néant
`Rect`	Rectangle	En haut à gauche (coordonnées x et y), en bas à droite (coordonnées x et y)
`circle`	Cercle	Centre (coordonnées x et y), rayon
`polygon`	Polygone	Chaque sommet (coordonnées x et y). le dernier couple de coordonnées devrait être identique au premier pour "fermer" le polygone.

Je suppose que vous restez dubitatif quant aux coordonnées de l'exemple précédent… Comment est-il possible de les déterminer simplement ? Deux solutions s'offrent à vous.

Création manuelle d'une image cliquable

La première méthode est la suivante :

1. Ouvrez votre programme de dessin habituel et chargez l'image concernée (ici, *franregs.gif*).
2. Placez votre curseur à l'emplacement approximatif du coin supérieur d'un rectangle, à l'emplacement de la région Poitou-Charentes. Pratiquement tous les programmes de dessin affichent quelque part les coordonnées actuelles du curseur.

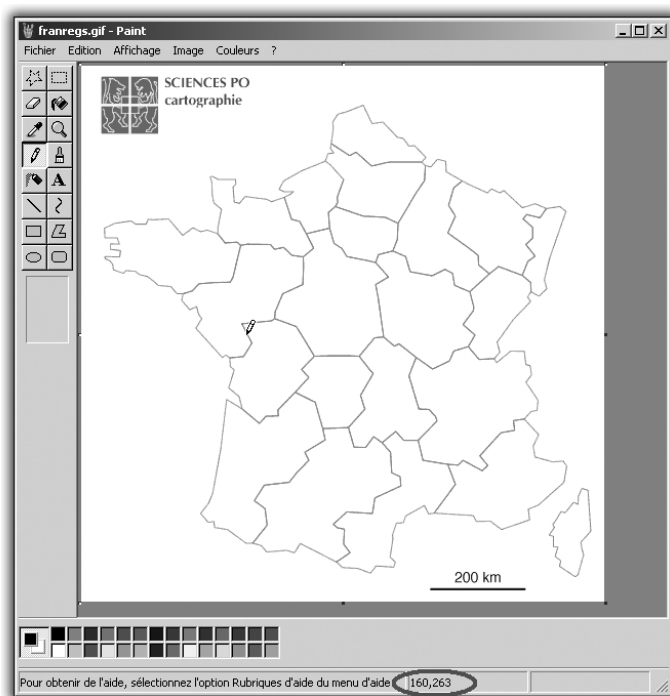


Figure 6.42 : *Les coordonnées de la position actuelle du curseur (ici encadrées d'un ovale) dans le logiciel de dessin Microsoft Paint*

3 Notez ces coordonnées, puis placez-vous sur le coin inférieur droit du rectangle pour la région Poitou-Charentes et notez à nouveau les coordonnées. Vous disposez des quatre paramètres nécessaires pour la définition d'une forme de type rectangle, que vous n'avez plus qu'à reporter dans le code :

```
<AREA shape="rect" coords="161,263,204,323"
      href="Poitou-Charentes.html"
      alt="R&eacute;gion Poitou-Charentes"
      title="Poitou-Charentes">
```

Remarquez que vous n'obtiendrez pas forcément tout à fait les mêmes coordonnées : quelques pixels de différence ne sont pas un problème, l'indication visuelle procurée à l'utilisateur étant essentiellement la modification de la forme du curseur (indiquant un lien) ainsi qu'éventuellement l'apparition de l'infobulle définie par l'attribut `title`.

4 Pour le cercle, placez-vous dans la région Île-de-France au centre du cercle, et notez ses coordonnées. Déplacez-vous

horizontalement ou verticalement pour définir le rayon souhaité et notez la différence entre coordonnées : c'est le rayon. Vous disposez des trois paramètres nécessaires pour la définition d'une forme de type cercle, que vous n'avez plus qu'à reporter dans le code :

```
<AREA shape="circle" coords="274,167,21"
      href="Ile de France.html"
      alt="R&eacute;gion Ile de France"
      title="Ile de France">
```

5 La démarche est un peu plus laborieuse pour un polygone : placez-vous sur un des angles de la région Centre et notez ses coordonnées. Déplacez-vous successivement sur les autres sommets du polygone souhaité et notez leurs coordonnées. Lorsque vous en avez terminé, reportez toutes ces coordonnées dans votre code. Ce pourrait être comme suit :

```
<AREA shape="polygon"
      coords="243,172,286,209,271,283,225,283,197,235,243,172"
     href="Centre.html"
      alt="R&eacute;gion Centre"
      title="Centre">
```

Remarquez que le dernier sommet doit être identique au premier. Si toutefois vous ne le spécifiez pas, les agents utilisateurs devraient être en mesure de le générer d'eux-mêmes.



Le rayon

Quand la valeur du rayon est un pourcentage, l'agent utilisateur devrait calculer le rayon final par rapport à la hauteur et à la largeur de l'objet associé. Le rayon devrait être la plus petite valeur des deux.

Création d'une image cliquable à l'aide d'un éditeur

La deuxième méthode consiste à recourir à un logiciel de création d'images cliquables. Il en existe un certain nombre, mais nous vous recommandons l'incomparable logiciel de dessin d'*Open Source* gratuit, The Gimp, qui dispose d'un tel utilitaire intégré.

1 Si vous n'en disposez pas déjà, téléchargez The Gimp depuis http://gimp-win.sourceforge.net/ et installez-le sur votre ordinateur. C'est ici la page Windows, mais il existe des versions Linux (la

version originelle) et Mac. Vous aurez également besoin de l'environnement d'exécution GTK +2 : n'oubliez pas de le télécharger et de l'installer également.

2 Lancez The Gimp et chargez l'image *Franceregs.gif*.

3 Choisissez **Filtres** > **Web** > **Image cliquable (imagemap)**.

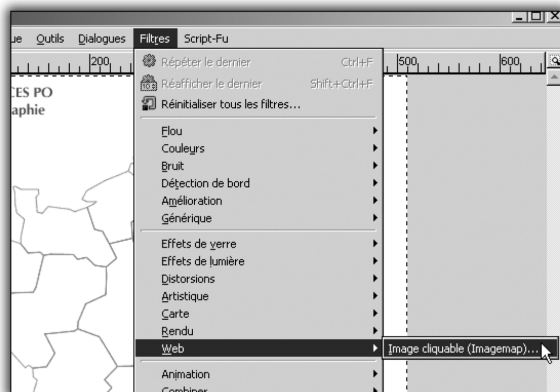


Figure 6.43 : *Création d'une image cliquable à l'aide de The Gimp*

4 Choisissez **Rectangle** dans le menu de gauche, puis dessinez un rectangle sur la région Poitou-Charentes .

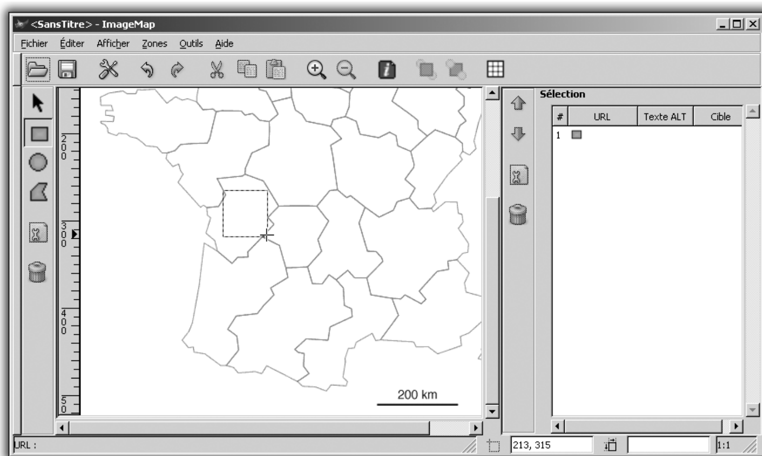


Figure 6.44 : *Utilisez le rectangle*

5 Renseignez les éléments nécessaires dans la fenêtre **Paramètres** de la zone qui s'ouvre, puis cliquez sur OK.

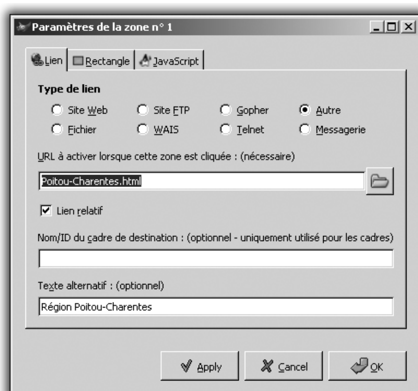


Figure 6.45 : *Fenêtre Paramètres de la zone*

6 Choisissez **Cercle** dans le menu de gauche, puis dessinez un cercle dans la région Île-de-France (attention : The Gimp trace un cercle en spécifiant le centre, puis en définissant le rayon). Renseignez les éléments nécessaires dans la fenêtre qui s'ouvre, puis cliquez sur OK.

7 Choisissez **Polygone** dans le menu de gauche, puis dessinez un polygone dans la région Centre. Renseignez les éléments nécessaires dans la fenêtre qui s'ouvre, puis cliquez sur OK. La fenêtre de The Gimp devrait afficher quelque chose de similaire à la figure suivante.

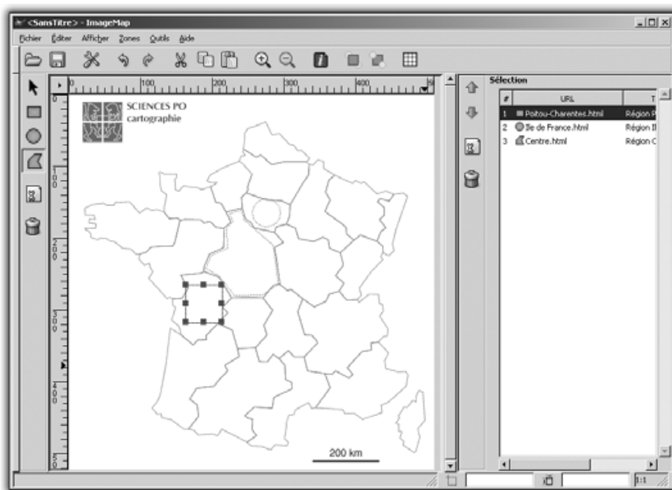


Figure 6.46 : *Image cliquable créée à l'aide de The Gimp*

Remarquez comme il est facile de définir une région polygonale à l'aide d'un tel outil !

8 Choisissez dans le menu **Afficher** > **Source**. Vous voyez le code source de l'image ainsi créée s'afficher. Vous pouvez copier-coller ce code (présenté en intégralité dans le listing suivant) dans votre fichier, pour y ajouter d'autres attributs, comme l'attribut `title`.

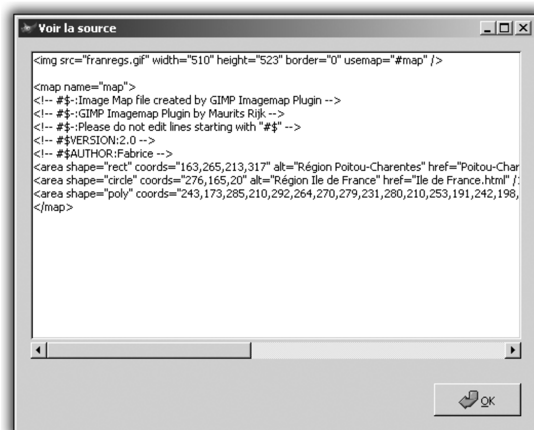


Figure 6.47 : Affichage du code source de l'image cliquable créée par The Gimp

Listing 6-1 : Code source de l'image cliquable créée par The Gimp

```
<img src="franregs.gif" width="510" height="523" border="0"
     usemap="#map" />
<map name="map">
<!-- #$-:Image Map file created by GIMP Imagemap Plugin -->
<!-- #$-:GIMP Imagemap Plugin by Maurits Rijk -->
<!-- #$-:Please do not edit lines starting with "#$" -->
<!-- #$VERSION:2.0 -->
<!-- #$AUTHOR:Fabrice -->
<area shape="rect" coords="163,265,213,317"
      alt="Région Poitou-Charentes"
      href="Poitou-Charentes.html" />
<area shape="circle"
      coords="276,165,20" alt="Région Ile de France"
      href="Ile de France.html" />
<area shape="poly" coords="243,173,285,210,292,264,270,
        279,231,280,210,253,191,242,198,224,221,214,220,
        186,229,177,224,166,240,161,243,169,246,173,245,171"
      alt="Région Centre" href="Centre.html" />
</map>
```

En examinant ce code et en le comparant à celui créé « manuellement » à l'aide de Paint, vous voyez que les coordonnées diffèrent légèrement.

Comme nous l'avons déjà souligné, cela ne présente pas la moindre importance. Remarquez également que le polygone est beaucoup plus complexe : nous avons suivi presque exactement les frontières de la région Centre. Par ailleurs, The Gimp a ajouté quelques commentaires, qu'il est de bon ton de conserver, tandis que son balisage respecte la spécification XHTML (balises `<IMG />` et `<AREA />`).

Recouvrement de zones

Si deux régions définies ou plus se chevauchent, le premier élément qui définit une région dans le document est prioritaire. Il est ainsi possible de spécifier des ancres pour créer des zones inactives dans une image cliquable. Dans l'exemple ci-dessous, la première ancre spécifie une petite région circulaire dépourvue de lien, tandis que la seconde spécifie une région circulaire de taille supérieure de même centre que la précédente. La combinaison des deux crée un anneau cliquable dont le centre est inactif et le bord actif. L'ordre des définitions des ancres est capital : le petit cercle doit être prioritaire par rapport au grand cercle.

```
<MAP name="carte1">
<P>
<A shape="circle" coords="100,200,50" title="Lien inactif"</A>
<A href="lien_anneau.html" shape="circle"
   coords="100,200,250" title="Lien actif".</A>
</MAP>
```

Vous pourriez également recourir à l'attribut `nohref` de l'élément `AREA` pour déclarer que la région ne possède pas de lien associé :

```
<A shape="circle" coords="100,200,50" nohref
   title="Lien inactif"</A>
```

Vous pourriez souhaiter doter la totalité de l'image d'un lien. Cela est possible en ajoutant un élément `AREA` dont l'attribut `shape` possède comme valeur `default`. Compte tenu de ce qui est exposé plus haut, ce dernier élément doit se trouver juste avant la balise de fermeture `</MAP>`, de façon que tous les éléments `AREA` précédemment définis soient prioritaires. Le code serait ici par exemple :

```
<AREA shape="default" HREF="francereg.html"
      alt="Carte de France par régions">
```

Les images cliquables sont largement employées pour proposer des barres de navigation ressemblant à tout sauf à une barre. Le recours à un graphisme plutôt qu'à du texte permet d'employer des polices de caractères « exotiques ». C'est une méthode largement employée sur les sites destinés aux enfants.

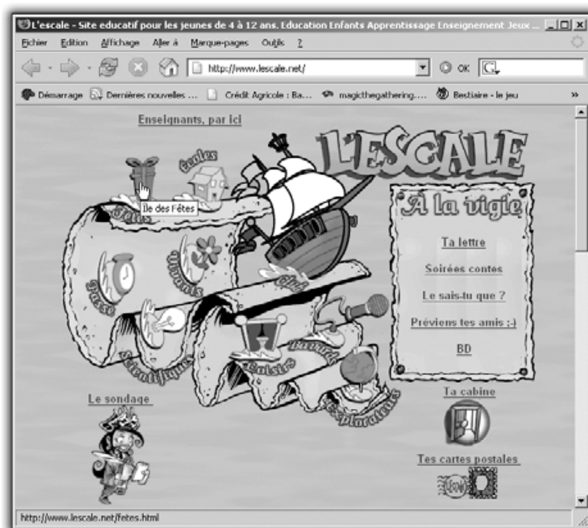


Figure 6.48 : *Exemple de site destiné à des enfants et recourant à une image cliquable pour la navigation*

En conclusion, l'élément `MAP` peut être associé à plusieurs éléments outre `IMG` : `OBJECT` et `INPUT`. Il est associé à un élément *via* l'attribut `usemap` de celui-ci. Il peut s'employer sans image associée pour des mécanismes de navigation généraux.

Le recours aux éléments `AREA` n'est toutefois pas la seule possibilité d'emploi de l'élément `MAP`. Le modèle de contenu de celui-ci permet en effet d'y placer :

- Un ou plusieurs éléments `AREA`. Ces éléments, dépourvus de contenu, spécifient les régions géométriques de l'image cliquable et les liens qui sont associés à chaque région. Les agents utilisateurs ne restituent pas général les éléments `AREA` : mieux vaut fournir un texte de remplacement pour chaque élément `AREA` à l'aide de l'attribut `alt`.
- Un contenu de type bloc composé entre autres d'éléments `A` spécifiant les régions géométriques de l'image cliquable et le lien associé à chaque région. Les agents utilisateurs devraient restituer le contenu de type bloc d'un élément `MAP` : cette méthode sert à créer des documents plus accessibles.
- Un contenu mixte d'éléments `AREA` et d'éléments `A`. En ce cas, les agents utilisateurs doivent ignorer les éléments `AREA`.

En mélangeant les contenus, les agents utilisateurs anciens prendront en charge les cartographies spécifiées par les éléments AREA, tandis que les agents utilisateurs récents tireront profit des contenus de type bloc plus étoffés.

Pensez à toujours offrir des alternatives textuelles aux images cliquables, au cas où les graphismes ne seraient pas disponibles ou les utilisateurs ne pourraient y accéder. Les agents utilisateurs peuvent employer le texte de l'attribut alt pour créer des liens textuels à la place des images cliquables graphiques. De tels liens peuvent être activés de diverses façons (clavier, commande vocale, etc.).

Sachez enfin que l'élément MAP ne présente pas de compatibilité descendante avec les agents utilisateurs HTML 2.0 : il sera ignoré par de tels agents utilisateurs.

Images cliquables côté serveur

Plutôt qu'une image côté client, il est possible de créer une image cliquable côté serveur. Les images cliquables côté serveur peuvent se révéler intéressantes lorsque l'image cliquable s'avère trop compliquée à réaliser côté client, mais font peser un poids supplémentaire sur le serveur, ce qui peut être diversement apprécié.

Lorsqu'une image cliquable côté serveur est définie pour un élément IMG, celui-ci doit se trouver dans un élément A et posséder l'attribut booléen ismap.

```
<P><A href="http://www.acme.com/cgi-bin/test">
        <IMG src="jeu.gif" ismap alt="Cible"></A>
```

La différence fondamentale avec une image cliquable côté client est que la définition des régions (effectuée dans le cas précédent avec l'élément MAP) est ici effectuée par un script en ASCII situé sur le serveur. Le contenu du fichier de script *test.map* cité dans l'exemple ci-dessus pourrait être quelque chose comme ceci :

```
default ../index.html 0, 0, 100, 460
rect ../index.html 0, 1, 100, 65
rect ../service1.html 0, 72, 100, 142
rect ../service2.html 0, 146, 100, 213
rect ../formation.html 0, 219, 100, 283
rect ../profils.html 0, 288, 100, 351
rect ../references.html 0, 355, 100, 403
rect ../contact.html 0, 407, 100, 453
```

Les images cliquables côté serveur ont été développées à l'origine par deux organismes, le NCSA et le CERN. Malgré des efforts certains de standardisation, il subsiste quelques petites différences entre eux, ainsi qu'au sein de leurs mises en œuvre. Vous devez vérifier le standard employé par le serveur Web avant de créer un script d'image cliquable côté serveur. L'exemple précédent est au format NCSA, mais voici la même chose au format CERN :

```
rect (0,1) (100,65) ../index.html
rect (0,72) (100,142) ../service1.html
rect (0,146) (100,213) ../service2.html
rect (0,219) (100,283) ../formation.html
rect (0,288) (100,351) ../profilse.html
rect (0,355) (100,403) ../references.html
rect (0,407) (100,453) ../contact.html
default ../index.html
```

Lorsque l'utilisateur active un lien en cliquant sur l'image, les cordonnées de ce clic à l'écran sont directement transmises au serveur qui héberge le script. Les coordonnées à l'écran sont exprimées par des valeurs en pixels d'écran relatives à l'image.

Pour transmettre au serveur le point cliqué, l'agent utilisateur dérive un nouvel URI à partir de l'URI spécifié par l'attribut `href` de l'élément `A`, en lui rajoutant à la fin le caractère « `?` » suivi des coordonnées « *x* » et « *y* », séparées par une virgule. Le lien est alors suivi en utilisant le nouvel URI. Par exemple, dans l'exemple donné, si l'utilisateur clique au point « `x=10, y=27` », l'URI dérivé sera « `http://www.acme.com/cgi-bin/test?10,27` ».

Les agents utilisateurs ne disposant d'aucun moyen de sélectionner des coordonnées spécifiques (par exemple, un agent utilisateur non graphique qui s'appuie sur une saisie au clavier, un agent utilisateur vocal, etc.), doivent envoyer les coordonnées « `0,0` » au serveur lors de l'activation du lien.

En conclusion, mieux vaut ne pas créer d'images cliquables côté serveur : celles-ci sont complètement inaccessibles aux utilisateurs de divers agents utilisateurs comme les navigateurs texte seul et les moteurs de recherche comme Google. Si vous devez absolument employer des images cliquables côté serveur, ajoutez une barre de navigation en texte seul en dessous. Celle-ci doit inclure de vrais liens texte vers chacune des pages auxquelles vous pourriez accéder en cliquant sur l'image cliquable.

Une image cliquable côté serveur ne peut être définie que pour des éléments IMG et INPUT, ce dernier devant alors être de type « image ».

Pour terminer, si l'élément IMG (dont les attributs sont résumés dans le tableau suivant) reste présent en HTML 4.01, il y est toutefois déconseillé en raison de ses limites : en effet, il ne répond pas au problème plus général de l'inclusion des types de médias nouveaux ou à venir et entraînent des problèmes d'accessibilité. Il convient désormais de lui préférer l'élément OBJECT.

Tableau 6.3 : *Attributs de l'élément IMG*

Attribut	Valeur(s)	Effet
src	URI	Fournit l'emplacement de l'image. Cet attribut est obligatoire.
align	center, left, right	Place l'image respectivement au centre, à droite ou à gauche. Peut autoriser le déroulement du texte autour du graphisme.
border	Nombre	Taille du cadre de l'image, exprimée en pixels (0 pour pas de cadre).
alt	Texte de description	Procure une alternative textuelle au graphisme.
usemap	Valeur de l'attribut name d'un élément MAP associé	Signale que cette image est une image cliquable côté client, définie par l'élément MAP associé.
ismap	Néant	Signale que cette image est une image cliquable côté serveur.

6.5. Inclusion générique d'images : élément OBJECT

Pour inclure des images, les auteurs devraient désormais privilégier l'élément OBJECT. Il s'agit d'un élément de type ligne, devant obligatoirement posséder une balise d'ouverture et une balise de fermeture, contrairement à IMG qui ne possédait pas de balise de fermeture.

En effet, la plupart des agents utilisateurs possèdent des mécanismes intégrés pour la restitution des types de données communs, tels que le

texte, les images GIF, les couleurs, les polices et une poignée d'éléments graphiques. Pour restituer les types de données qu'ils ne reconnaissent pas nativement, les agents utilisateurs lancent en général des applications externes. L'élément `OBJECT` permet aux auteurs de mieux contrôler si les données doivent être restituées de manière externe ou par un certain programme, spécifié par l'auteur, qui restitue ces données au sein de l'agent utilisateur.

Voici comment employer l'élément `OBJECT` pour inclure une image.

1 Ouvrez le Bloc-Notes, puis ouvrez le fichier *region1_6_1.html*.

2 Modifiez d'abord la version du document :

```
<META name ="version" content="1.6.2">
```

3 Remplacez l'élément `IMG` par ce qui suit :

```
<OBJECT data="Images/coucher de soleil.jpg"
        type="image/gif" width="267" height="200"
        title="Un coucher de soleil sur le lac de Linciel">
</OBJECT>
```

4 Enregistrez votre fichier sous le nom *region1_6_2.html*, puis examinez-le dans votre navigateur.



Figure 6.49 : *Inclusion d'image à l'aide de l'élément OBJECT*

Visuellement, le recours à l'élément `OBJECT` plutôt qu'à l'élément `IMG` ne produit aucune différence (reportez-vous à la figure de la page 242 pour comparer). Revenez au code pour l'examiner à nouveau. Le nouvel élément possède deux nouveaux attributs :

- L'attribut `data` de l'élément `OBJECT` remplace l'élément `src` de l'élément `IMG` : c'est un URI, pouvant être interprété relativement à la valeur de l'attribut optionnel `codebase` et qui l'est par défaut relativement à l'URI de base du document.
- L'attribut `type` spécifie le type des données spécifiées par l'URI de l'attribut `data`. Cet attribut facultatif est toutefois recommandé : il évite le chargement par l'agent utilisateur d'informations dont il ne peut reconnaître le type de contenu. Si la valeur de cet attribut diffère de l'en-tête HTTP *Content-Type* renvoyé par le serveur quand l'objet est renvoyé, c'est l'en-tête HTTP *Content-Type* qui a priorité.

Remarquez également que nous avons supprimé l'attribut `alt` et sa valeur. En effet, une des caractéristiques les plus intéressantes de l'élément `OBJECT` est qu'il fonctionne un peu comme la classique instruction *si…sinon* (*if… else*). Autrement dit, il peut posséder un contenu interprété uniquement s'il est impossible d'interpréter le contenu de sa balise d'ouverture. Pour disposer d'un texte alternatif affiché au cas où l'agent utilisateur ne peut interpréter l'image, procédez comme suit :

5 Modifiez l'élément `OBJECT` en insérant le texte ci-dessous juste avant la balise de fermeture `</OBJECT>` :

```
<OBJECT data="Images/coucher de soleil.jpg"
        type="image/gif" width="267" height="200"
        title="Un coucher de soleil sur le lac de Linciel">
<!—Si l'image ne peut être affichée, afficher le texte -->
Coucher de soleil sur le lac de Linciel
</OBJECT>
```

6 Enregistrez à nouveau votre fichier, sous le même nom.

En l'examinant dans votre navigateur, vous ne constaterez aucune différence. Cette syntaxe semble n'être qu'une nuance, mais elle prend toutefois tout son sens lorsque vous savez qu'il est possible d'imbriquer des éléments `OBJECT`, ainsi que d'employer ceux-ci pour spécifier bien d'autres types de contenus, dont des scripts. Prenons un exemple.

1 Ouvrez le fichier *FranceReg2.html*.

2 Modifiez d'abord la version du document :

```
<META name ="version" content="1.6.3">
```

3 Modifiez comme suit le contenu de la première cellule du tableau. Attention, veillez à bien refermer les éléments OBJECT que vous allez ajouter à la fin du fragment du code :

```
<TD rowspan="3">
<!-- Essayer d'abord un applet en Python -->
<OBJECT title="Coucher de soleil sur le lac de Linciel"
     classid="Linciel.py">
  <!-- Sinon, essayer l'animation MNG -->
  <OBJECT data="Linciel.mng" type="application/mng">
    <!-- Sinon, essayer l'image JPEG -->
    <OBJECT data="Images/coucher de soleil.jpg"
       type="image/gif" width="267" height="200"
       title="Un coucher de soleil sur le lac de Linciel">
    <!-- Sinon, le texte en dernier recours -->
    Coucher de soleil sur le lac de Linciel
    </OBJECT>
  </OBJECT>
</OBJECT>
</TD>
```

4 Enregistrez votre fichier sous le nom *region1_6_3.html*, puis examinez-le dans votre navigateur.

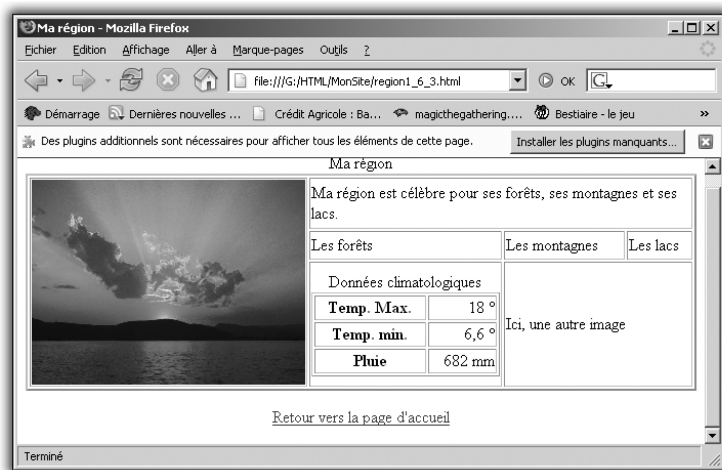


Figure 6.50 : *Possibilités alternatives à l'aide de l'élément OBJECT*

Le navigateur étant pour le moment incapable d'interpréter un script Python ou une animation MNG (souvenez-vous : c'est le format d'animation dérivé de PNG), il affiche la première possibilité reconnue, ici l'image JPEG. Remarquez que Firefox signale par une barre de message que cette page nécessite des modules complémentaires (*plug-*

ins) et propose d'essayer de les installer. Nous reviendrons dans le Chapitre 10, consacré aux scripts, sur l'emploi de l'élément OBJECT avec les scripts.

Vous voyez ici une partie de la souplesse fantastique offerte par cet élément. Le reste ne dépend que de votre imagination.

Bien évidemment, cet élément permet également l'insertion d'images cliquables, côté client ou côté serveur. Commençons par une image cliquable côté client.

1 Revenez au fichier *region1_6_2.html*.

2 Modifiez d'abord la version du document :

```
<META name ="version" content="1.6.3">
```

3 Modifiez le code comme suit. Attention, n'oubliez pas de refermer l'élément OBJECT après la fermeture de l'élément MAP !

```
<P>
<OBJECT data="Images/franregs.gif" datatype="image/GIF"
   usemap="#freg">
<MAP name="freg">
<P>Quelques précisions :
  <A href="Poitou-Charentes.html" shape="rect"
     coords="161,263,204,323" title="Poitou-Charentes">
       R&eacute;gion Poitou-Charentes</A>
  <A href="Ile de France.html" shape="circle"
     coords="274,167,21" title="Ile de France">
       R&eacute;gion Ile de France</A>
  <A href="Centre.html" shape="polygon"
     coords="243,172,286,209,271,283,225,283,
     197,235,243,172" alt=" " title="Centre">
       R&eacute;gion Centre</A>
</MAP>
</OBJECT></P>
```

Remarquez que l'élément MAP est « dissimulé » dans le contenu de l'élément OBJECT. Nous ne voulons pas restituer le contenu de l'élément MAP lorsque l'élément OBJECT est restitué. Il ne le sera que si l'élément OBJECT ne peut l'être.

4 Enregistrez votre fichier sous le nom *FranceReg3.html*, puis examinez-le dans votre navigateur (voir Figure 6.51).

Cette figure ne diffère en apparence en rien de la version IMG (reportez-vous à la figure de la page 242). Nous allons simuler un agent utilisateur incapable d'interpréter l'image, en modifiant le nom et le type de l'image :

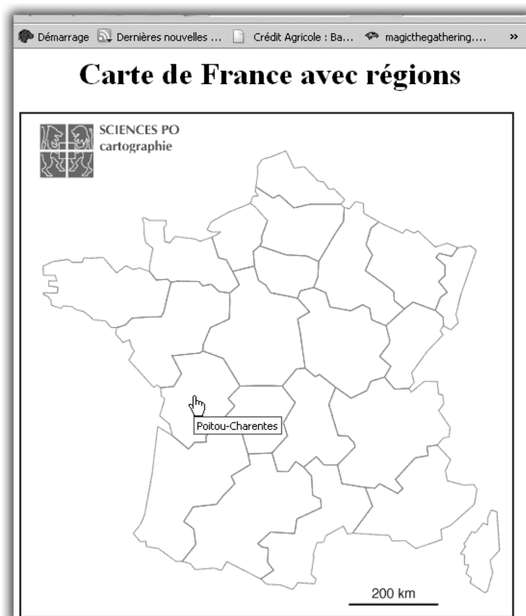


Figure 6.51 : *Image cliquable à l'aide de l'élément OBJECT*

5 Modifiez comme suit l'élément OBJECT :

```
<OBJECT data="Images/franregs.mng"
   datatype="image/MNG" usemap="#freg">
```

6 Enregistrez votre fichier sous le nom *FranceReg3err.html*, puis examinez-le dans votre navigateur.



Figure 6.52 : *Alternative proposée grâce à un élément OBJECT*

L'image ne pouvant être affichée, vous voyez à la place un menu textuel.

Vous pourriez souhaiter que l'image cliquable soit restituée même si l'agent utilisateur est incapable de restituer l'élément OBJECT. Par exemple, en associant une image cliquable à un élément OBJECT et en plaçant une barre de navigation textuelle en bas de page. Pour cela, placez l'élément MAP en dehors de l'élément OBJECT :

Listing 6-2 : Elément MAP restitué

```
<HTML>
...
  <BODY>
    <H1 align="center">Carte de France avec régions</H1>
    <P>
      <OBJECT data="Images/franregs.mng"
          datatype="image/MNG" usemap="#freg">
      </OBJECT>
    </P>
    ... reste du corps
    <MAP name="freg">
      <P>Quelques précisions :
        <A href="Poitou-Charentes.html" shape="rect"
           coords="161,263,204,323"
           title="Poitou-Charentes">
           R&eacute;gion Poitou-Charentes</A>
        <A href="Ile de France.html" shape="circle"
           coords="274,167,21" title="Ile de France">
           R&eacute;gion Ile de France</A>
        <A href="Centre.html" shape="polygon"
           coords="243,172,286,209,271,283,225,
             283,197,235,243,172" title="Centre">
           R&eacute;gion Centre</A>
    </MAP>
  </BODY>
</HTML>
```

La figure suivante montre l'aspect de cette page dans un navigateur (voir Figure 6.53).

Comme avec l'élément IMG, vous pouvez créer une image cliquable en employant des éléments AREA.

1 Ouvrez le fichier *FranceReg.html*.
2 Modifiez d'abord la version du document :

```
<META name ="version" content="1.6.4">
```

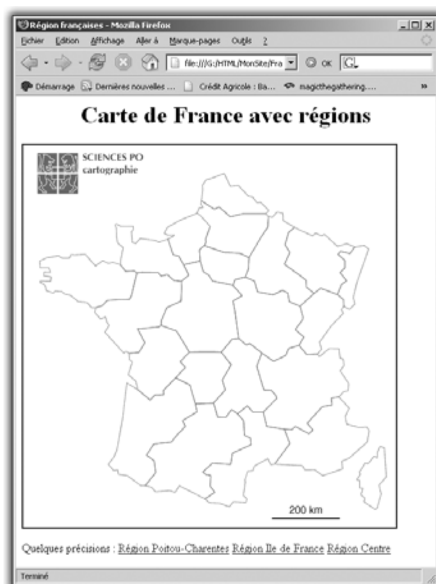


Figure 6.53 :
Affichage simultané de l'image cliquable et de liens textuels

3 Remplacez l'élément `IMG` par un élément `OBJECT`, comme suit :

```
<P>
<OBJECT data="Images/franregs.gif" datatype="image/GIF"
✂ usemap="#freg">
<P> Carte des France avec r&eacute;gions
</OBJECT>
</P>
```

4 Enregistrez votre fichier sous le nom *FranceReg4.html*, puis examinez-le dans votre navigateur (voir Figure 6.54).

Une fois encore, cette figure ne diffère apparemment en rien de la version `IMG` (reportez-vous à la figure de la page image06_34).

La présence de l'attribut `usemap` sur un élément `OBJECT` implique que l'objet inclus soit une image cliquable côté client associée. En ce cas, l'agent utilisateur peut produire une interaction utilisateur avec cet élément `OBJECT`, uniquement en fonction de l'image cliquable côté client. Ceci permet à certains agents utilisateurs comme les navigateurs vocaux ou les robots d'interagir avec l'élément `OBJECT` sans devoir le traiter. L'agent utilisateur peut même choisir de ne pas ramener (ou traiter) l'objet. Lorsqu'un élément `OBJECT` possède une image cliquable associée, vous ne devez pas supposer que l'objet sera traité par tous les agents utilisateurs.

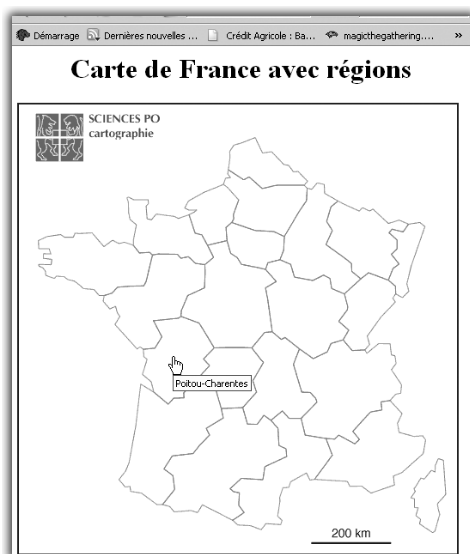


Figure 6.54 : *Image cliquable à l'aide de l'élément OBJECT et d'éléments AREA*

L'élément `OBJECT` peut aussi apparaître comme contenu de l'élément `HEAD`. Les agents utilisateurs ne restituant généralement pas les éléments placés dans la section `HEAD`, veillez à ce qu'aucun élément `OBJECT` situé dans la section `HEAD` ne spécifie un contenu qui peut être restitué.

Le Chapitre 7, traitant des jeux d'encadrement, propose un exemple d'inclusion d'élément `OBJECT` dans l'élément `HEAD`. De même, le Chapitre 9, traitant des formulaires, procure quelques précisions sur l'emploi d'éléments `OBJECT` dans les formulaires.

Tableau 6.4 : *Principaux attributs de l'élément OBJECT relatifs aux images*

Attribut	Valeur(s)	Effet
`data`	Uri	Spécifie la localisation des données de l'objet, par exemple les données d'images pour les objets définissant des images ou, plus généralement, la forme sérialisée d'un objet qui peut être utilisée pour recréer cet objet.
`type`	type-de-contenu	Facultatif. Spécifie le type de contenu des données spécifiées par l'attribut `data`.
`usemap`	Néant	Signale que l'objet associé est une image cliquable.

Tableau 6.5 : *Attributs de l'élément OBJECT*		
Attribut	**Valeur(s)**	**Effet**
classid	Uri	Spécifier la localisation de l'implémentation d'un objet *via* un URI. Peut être employé en combinaison ou en remplacement de l'attribut data, selon le type de l'objet impliqué.
codebase	Uri	Spécifie le chemin de base qui sert à la résolution des URI relatifs spécifiés par les attributs classid, data et archive. Quand il est absent, sa valeur par défaut correspond à l'URI de base du document courant.
codetype	type-de-contenu	Spécifie le type de contenu des données attendues lors du chargement de l'objet spécifié par l'attribut classid. Cet attribut est optionnel, mais recommandé lorsque l'attribut classid est spécifié. Il permet en effet à l'agent utilisateur d'éviter le chargement des informations dont les types ne sont pas reconnus. Quand il est absent, sa valeur par défaut est celle de l'attribut type.
data	Uri	Spécifie la localisation des données de l'objet, par exemple les données d'images pour les objets définissant des images, ou plus généralement, la forme sérialisée d'un objet qui peut être utilisée pour recréer cet objet.
type	type-de-contenu	Facultatif. Spécifie le type de contenu des données spécifiées par l'attribut data.
archive	liste-uri	Spécifie une liste d'URI, séparés par des espaces, des archives contenant les ressources concernant l'objet. Cette liste peut inclure les ressources spécifiées par les attributs classid et data. Le préchargement des archives diminue généralement le temps de chargement des objets. Les archives spécifiées comme URI relatifs devraient être interprétées relativement à l'attribut codebase.
declare		Quand il est présent, la définition de l'élément OBJECT courant n'est qu'une déclaration. L'objet doit être instancié par la suite *via* une définition OBJECT qui se réfère à cette déclaration.

Tableau 6.5 : *Attributs de l'élément OBJECT*		
Attribut	**Valeur(s)**	**Effet**
`standby`	texte	Spécifie le message que l'agent utilisateur peut restituer pendant le chargement de l'implémentation et des données de l'objet.

Diminution du temps de téléchargement

Comme cela a déjà été souligné, le temps de chargement d'une page est proportionnel au nombre et à la taille des graphismes employés sur cette page. L'emploi des illustrations doit donc être judicieusement mesuré.

Heureusement, plusieurs méthodes permettent d'améliorer le temps de chargement d'une page, même si elle contient des illustrations.

Réduction de la taille physique des images

La première méthode consiste à employer des images de plus faible taille, ajustées à la résolution envisagée pour la page Web. Une erreur fréquente consiste à employer des fichiers image « tels quels » et à modifier la taille d'affichage à l'écran. Cette erreur devient de plus en plus fréquente avec le développement de la photo numérique, dont le pouvoir de résolution augmente d'année en année, mais elle se rencontre aussi lors de l'emploi de documents numérisés à l'aide d'un scanner.

Il y a parfois de quoi se perdre dans la jungle des unités de mesure employées par les dispositifs d'acquisition d'images (scanneurs, appareils photo et caméras numériques) et de restitution (imprimantes, tables traçantes, flasheuses).

Le terme le plus employé est celui de dpi (*dots per inch*) : c'est le nombre de points sur un pouce, soit 2,54 cm. Ce terme est employé pour les scanneurs et les imprimantes. De nos jours, une résolution d'impression de 600 dpi est un minimum, une résolution de 2 400 dpi n'offrant plus rien d'exceptionnel. Sachez cependant que les limites sont proches : les professionels de l'imprimerie ne dépassent que rarement 3 600 dpi pour des reproductions d'art, en employant des papiers spéciaux fort onéreux. Une résolution supérieure n'apporte rien et peut même dégrader le rendu final.

Les imprimantes noir et blanc (comme les imprimantes laser « ordinaires ») fonctionnent en tons de gris : chaque pixel de l'image de départ est traduit en carré de 16 pixels (4 × 4) afin de pouvoir afficher des teintes différentes. La résolution est donc en fait divisée par 4 par rapport à un dessin au trait (bien qu'elle puisse être augmentée de façon logicielle par l'imprimante). Cette organisation se faisant horizontalement, l'appellation correcte devrait être lpi (ligne par pouce) au lieu de dpi.

En revanche, les appareils photo numériques parlent le plus souvent en nombre total de pixels (4, 5 ou 6 millions pour les modèles les plus perfectionnés), tandis que les écrans mélangent la notion de diagonale de l'écran (15, 17 ou 19 pouces) et de résolution (800 × 600, 1024 × 768, 1280 × 960 ou 1280 × 1024, etc.). Pas facile de comparer, surtout en tenant compte du fait que, pour un moniteur cathodique, l'écran affiché est généralement plus petit que la taille physique de l'écran !



Appareils photo numériques : 4 millions de pixels, c'est beaucoup !

Oui et non … Grossièrement et traditionnellement, une photo présente un rapport largeur/hauteur de 3/2 (en mode paysage, c'est-à-dire horizontalement. C'est l'inverse en mode portrait). Autrement dit, vous avez :

```
résolution = 3 l x 2 l
```

D'où

```
l = (résolution/6)½
```

Avec une résolution de 4 millions de pixels, vous obtenez :

```
l = (4 000 000/6)½ = (2 666 666)½ = 816,5 pixels
```

Une photo obtenue avec un appareil photo d'une résolution de 4 millions de pixels possède donc une largeur d'environ 2 449 pixels et une hauteur de 1 633 pixels. Autrement dit, imprimée avec une imprimante atteignant une résolution de 1 200 points par pouce, pour conserver cette résolution, la photo obtenue mesurerait environ 2 pouces par 1,4 pouces, soit 5 cm par 3,5 cm… Pas très grand !

En revanche, si vous imprimez avec une résolution de 350 dpi (par exemple), la photo atteindra 17,7 × 12 cm : c'est plus raisonnable.

Comprenez bien toutefois que si, partant de cette photo de 4 millions de pixels placée dans un logiciel et ayant défini une taille d'impression de 17,7 × 12 cm, même en sélectionnant une résolution d'impression de 1 200 dpi, la vraie résolution de la photo imprimée (pas celle de l'impression elle-même) ne sera que de 350 dpi ! Les points supplémentaires ont été obtenus par extrapolation…

Mieux vaut donc convertir cette résolution théorique en nombre de points : si vous considérez une photo normale de format 24 × 18 cm, soit 9,6 × 7,2 pouces, avec une résolution de 1 200 points par pouce vous avez imprimé (9,6 × 1 200) × (7,2 × 1 200) = 99 532 800 points : plus de 99,5 millions de points ! Nous sommes loin des 4 millions de pixels de la photo obtenue avec l'appareil photo numérique…

Le problème est toutefois totalement différent si vous affichez votre photo numérique à l'écran : avec la résolution actuelle la plus fréquente de 1 024 × 768, une telle photo occuperait une superficie équivalente à presque 5 écrans (presque 2,4 écrans en largeur et un peu plus de 2 en hauteur). Elle est donc totalement inexploitable en l'état sur une page Web.

Tableau 6.6 : *Résolution et part de marché des écrans mondiaux en 2006 (source : OneStatjuin2006)*

Résolution	**Nombre total de points**	**Résolution en dpi (selon la taille de l'écran, de 15 à 19')**	**Part de marché 2006**
1 024 × 768	786 432	73 à 81 dpi	56,1 %
1 280 × 1 024	1 310 720	91 à 108 dpi	15,8 %
800 × 600	480 000	57 à 62 dpi	12,0 %
1 152 × 864	949 248	82 à 91 dpi	54,0 %

Le problème est similaire lorsque vous voulez placer sur votre site une image provenant d'un scanneur à plat. Si la numérisation est effectuée dans le but d'une impression, vous devez choisir une résolution d'entrée égale au produit de la résolution de sortie par le rapport de taille souhaité par rapport à l'original. Par exemple, avec une imprimante réglée sur 600 dpi et un rapport d'impression du double de l'original, vous aurez : 600 × 2 = 1200 dpi.

En revanche, en vue d'un affichage écran, une résolution d'entrée de 75 dpi est largement suffisante, voire exagérée, puisqu'il est exceptionnel qu'une image occupe tout l'écran : tout dépend encore une fois du rapport de taille entre l'original et ce que vous souhaitez obtenir à l'écran.

Nous n'irons pas plus loin dans l'étude des scanneurs et autres périphériques d'acquisition ou de restitution d'images, ce thème sortant du champ de cet ouvrage.

En reprenant toutefois notre exemple d'image numérique, celle-ci codée en vraies couleurs (32 bits par pixel) devrait occuper 32 × 4 millions = 128 millions de bits, soit, converti en octets, environ 16 Mo…

Heureusement, les principaux formats de fichiers graphiques opèrent une compression des données aboutissant à une réduction de la taille des fichiers.

Réduction de la taille des fichiers

Vous savez toutefois probablement que les fichiers graphiques peuvent être trouvés sous différentes formes, reconnaissables à l'extension du nom de fichier : JPG, GIF, PNG, TIF, BMP, etc., et que ces formats réalisent une compression logicielle des données plus ou moins efficace.

À titre d'exemple, le tableau suivant présente une comparaison de la taille des fichiers de l'image exemple de *Coucher de soleil* employée dans notre page *Région*, qui mesure 800 × 600 pixels, selon le format de fichier employé.

Tableau 6.7 : *Comparaison de quelques formats de fichiers graphiques*

Type	Nbre bit/pixel	Couleurs	Taille en mémoire	Taille réelle du fichier
JPG	24	16 millions	1 406 Ko	70 Ko
PNG	24	16 millions	1 406 Ko	532 Ko
TIF	24	16 millions	1 406 Ko	1 254 Ko
BMP	24	16 millions	1 406 Ko	1 407 Ko
TIF	16	64 000	703 Ko	551 Ko
PNG	16	64 000	703 Ko	378 Ko
TIF	8	256	469 Ko	194 Ko
PNG	8	256	469 Ko	163 Ko

Tableau 6.7 : Comparaison de quelques formats de fichiers graphiques				
Type	**Nbre bit/pixel**	**Couleurs**	**Taille en mémoire**	**Taille réelle du fichier**
GIF	8	256	469 Ko	251 Ko
TIF	4	16	234 Ko	59 Ko
PNG	4	16	234 Ko	49 Ko
GIF	4	16	234 Ko	49 Ko

Remarquez tout d'abord que la taille en mémoire correspond bien au calcul théorique :

```
800 × 600 × 24 = 11 524 000 bits = 1 440 000 octets = 1 406 Ko
```

Cette résolution ne permet pas le format GIF, qui n'accepte pas plus de 256 couleurs. En revanche, le format BMP la reconnaît : comme vous le voyez, il n'effectue aucune compression. Ce format n'est d'ailleurs pas reconnu sur le Web, pas plus que le format TIF : ils ne sont là que pour permettre la comparaison. Dans cet exemple, le fichier JPEG est de loin le plus petit.

La première possibilité pour réduire la taille de ce fichier sans modifier le format de l'image consiste à réduire le nombre de couleurs, pour passer à 64 000 couleurs. Cela élimine le format JPEG, obligatoirement en 16 millions de couleurs. Nous restons toutefois loin de la taille du fichier JPEG, malgré cette baisse de qualité. Poursuivons en passant en 256 couleurs, pour pouvoir tester le format GIF.

Si les fichiers restent de taille supérieure à celui d'un fichier JPEG en 16 millions de couleurs, le format GIF se révèle le plus gourmand parmi les autres. Poussons le raisonnement à son terme en réduisant radicalement le nombre de couleurs à 16 (soit 4 bits par couleur).

La taille des fichiers poursuit sa réduction, passant cette fois en dessous de celle du fichier JPEG. Il est toutefois permis de s'interroger sur la pertinence d'une telle réduction de qualité, pour ne gagner que 30 % en taille !

Sachez toutefois que les chiffres présentés ici sont propres à cette image : une autre image pourrait donner des résultats différents, le format GIF pouvant s'avérer plus intéressant en 256 couleurs. Il reste

cependant important de noter que, inversement, la conversion d'un fichier GIF en JPEG apporte parfois une grande amélioration !

Tout dépendant des caractéristiques du fichier, seuls les essais et l'expérience pourront vous procurer un indice.

Il existe par ailleurs plusieurs programmes permettant de réduire la taille des images JPEG avec une perte de qualité minimale. C'est le cas de Paint Shop Pro.

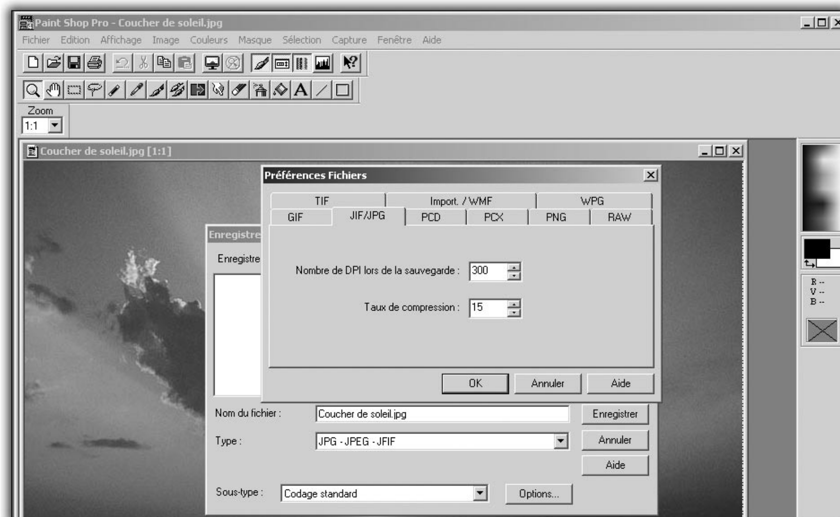


Figure 6.55 : *Paint Shop Pro permet de réduire la taille d'une image JPG*

SmartSaver 3.0, conçu par Ulead, est plus facile à employer. Vous pouvez en télécharger une version d'évaluation depuis le site **www.ulead.com/webutilities/frwhere.htm** Cet utilitaire réalise un extraordinaire travail d'ajustement de compression en comparant les résultats selon les réglages et les types de fichiers. Deux fenêtres permettent de comparer l'original, à gauche, et la version compressée, à droite, en taille et en qualité. Les effets des différents réglages sautent immédiatement aux yeux : au prix d'une baisse de qualité perceptible, mais somme toute acceptable, la taille du fichier passe de 71 à 41 Ko ! (voir Figure 6.56)

SmartSaver propose des outils pour recadrer et modifier la taille des images : ainsi, tous les outils nécessaires pour améliorer le chargement d'une image sont-ils à la portée de main.

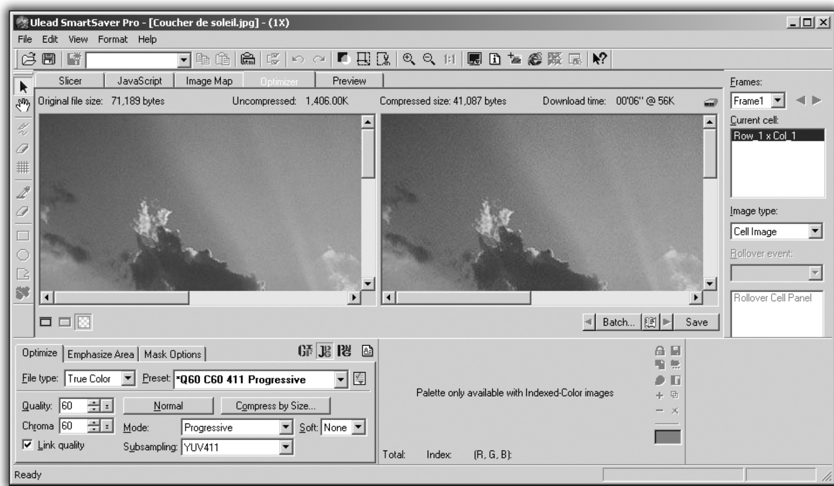


Figure 6.56 : *Réduction de la taille d'une image avec SmartSaver 3.0*

Recours à une page de vignettes

Pour afficher un groupe d'images de qualité photographique, une solution astucieuse consiste à créer une page qui présente des réductions de ces images, appelées vignettes ou miniatures (*thumbnail*). Chaque vignette est reliée à l'image en taille réelle. Ceci permet aux visiteurs de voir ce qui est disponible avec une perte de temps minimale en chargement, et ils peuvent ainsi voir uniquement les images qui les intéressent.

6.6. Résumé

- La toute première étape de l'enjolivement d'une page consiste à y ajouter de la couleur.
- Exploiter les couleurs est de la plus grande simplicité en HTML, grâce à divers attributs de l'élément `BODY`.
- La valeur hexadécimale employée pour spécifier une couleur porte le nom de *code RGB* de cette couleur.
- Vous modifiez la couleur du texte pour ou dans un élément grâce à l'attribut `color` d'un élément `FONT`. Vous modifiez la couleur

du fond des éléments d'un tableau à l'aide de l'attribut `bgcolor` de l'élément concerné.

- Les couleurs peuvent être désactivées par l'utilisateur. Tous les navigateurs ou presque possèdent désormais des options de gestion des couleurs ou autorisent les utilisateurs à définir un ensemble particulier de couleurs à utiliser pour tout site visité. Tous ces attributs sont désormais déconseillés, au profit des feuilles de style.
- Il est possible de placer des images sur un site Web. Les formats de fichiers graphiques les plus fréquents sont GIF (*Graphic Interchange Format*), JPEG (*Join Photographic Experts Group*) et PNG (*Portable Network Graphic*). Pour pouvoir diffuser sur votre site une image, vous devez avoir le droit de le faire : elle est libre de droits, vous avez obtenu l'accord de son propriétaire ou vous en êtes vous-même l'auteur.
- Vous insérez une image d'arrière-plan pour la totalité de la page dans l'élément `BODY` à l'aide de l'attribut `background`.
- Vous insérez une image dans un élément HTML à l'aide des éléments `IMG` et `OBJECT`. Ce dernier élément, plus puissant, est désormais à privilégier.
- Vous pouvez employer des images statiques ou des images animées (gifs animés). Vous pouvez créer vos propres images animées.
- Il est possible de créer et d'insérer des images cliquables : un clic sur une portion de l'image déclenche une action, souvent (mais pas toujours) l'ouverture d'une nouvelle page. Vous employez pour ce faire les éléments `MAP` et `AREA`, associés à des éléments `IMG` ou `OBJECT`.
- De façon générale, mieux vaut toujours chercher à diminuer le temps de téléchargement d'une page Web. Vous devez veiller à bien contrôler la taille de vos images. Un choix judicieux du format de fichier, ainsi que le recours à certains utilitaires, peut permettre de réduire sensiblement la taille des fichiers concernés, sans perte de qualité visuelle.

Chapitre 7

Jeux d’encadrement

Jeu d’encadrement : élément FRAMESET 245
Contenu d’un jeu d’encadrement : élément FRAME 248
Détermination du cadre cible : attribut target 261
Ensembles de cadres imbriqués 265
Partage de données entre cadres 269
Contenu de remplacement : élément NOFRAMES 270
Cadres en ligne : élément IFRAME 271
Travail avec les jeux d’encadrement 273
Résumé 275

Les cadres ou *frames* sont largement employés en HTML. Ils constituent le principal moyen d'afficher de multiples pages HTML sur le même écran au même moment. Vous pourriez ouvrir plusieurs fenêtres ou onglets sous votre navigateur, mais seriez alors contraint de basculer constamment de l'un à l'autre. Grâce aux cadres, il est possible de concevoir un site qui présente aux utilisateurs plusieurs fenêtres. Ces fenêtres d'information doivent cependant être liées : les utilisateurs ne doivent pas avoir l'impression de perdre une place précieuse pour des éléments sans importance.

Les cadres sont fréquemment employés pour contrôler la navigation de l'utilisateur sur le site. Des études ont démontré qu'une barre de navigation ou un menu n'étaient utiles qu'à condition d'être situés en haut d'un document. Beaucoup d'utilisateurs du Web ne font jamais défiler l'écran vers le bas : les éventuelles barres de navigation situées en bas d'une page doivent donc être des répliques de celles du haut. Mieux vaut recourir à un cadre pour placer définitivement les outils de navigation à un endroit bien défini de l'écran.

Des cadres permettent de présenter les documents selon des vues multiples : des fenêtres indépendantes ou des sous-fenêtres. C'est un moyen de garder visibles certaines informations, tandis que d'autres fenêtres défilent ou sont remplacées. Par exemple, dans la même fenêtre, un cadre pourrait afficher une bannière statique, un deuxième cadre afficher un menu de navigation et un troisième le document principal qui peut défiler ou être remplacé au gré de la navigation *via* le deuxième cadre.

Ce chapitre montre comment construire un jeu d'encadrement ou *frameset*, aussi nommé « ensemble de cadres ». Cela s'effectue par le biais d'un fichier qui définit les cadres, le mode d'ouverture d'une page dans un cadre en utilisant le lien présent dans un autre, et la façon d'utiliser les éléments de script pour que les cadres fonctionnent correctement. Vous allez apprendre à concevoir un ensemble de cadres attrayant, ergonomique et efficace.

En raison des problèmes de conception qui leur sont propres, les cadres sont une technique avancée de HTML et doivent être traités comme tels.

Éléments étudiés

```
FRAMESET, cols, rows
FRAME, name, src, noresize, scrolling, frameborder,
marginheight, marginwidth
target
BASE
NOFRAMES
IFRAME
```

7.1. Jeu d'encadrement : élément FRAMESET

Un fichier de définition de jeu d'encadrement possède une syntaxe analogue à celle d'un document HTML ordinaire, mais son code diffère. Tout d'abord, avec HTML 4.01, ces documents utilisent un élément `DOCTYPE` différent en tête de page. Ils sont en outre dépourvus d'élément `BODY`, remplacé par un élément `FRAMESET`.

Nous allons immédiatement mettre en jeu les cadres pour améliorer notre site Web. Le meilleur moyen de commencer consiste à tracer sur une feuille de papier l'aspect souhaité, en attribuant un nom à chaque cadre. Ici, comme le montre la figure suivante, notre ensemble de cadres ne contient que deux cadres. Le premier cadre, nommé `nav` (pour navigation), occupe la totalité de la largeur de l'écran et ne possède qu'une hauteur de 50 pixels. Le second cadre, nommé `princ` (pour principal) occupe le reste de la fenêtre.

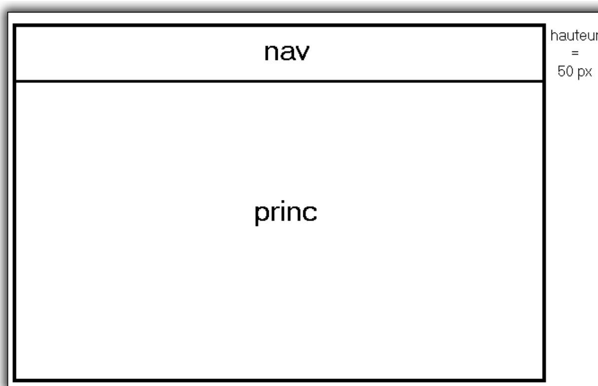


Figure 7.1 : *Schéma de l'ensemble de cadres de notre site*

Passons à la pratique.

1 Ouvrez le Bloc-Notes.

2 Saisissez l'élément `DOCTYPE` caractéristique d'un ensemble de cadres :

```
<!DOCTYPE HTML PUBLIC
   "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
```

3 Poursuivez comme avec n'importe quel document HTML, en saisissant les éléments HTML et HEAD. Comme à l'habitude, n'oubliez pas de refermer immédiatement ces éléments :

```
<HTML>
<HEAD>
</HEAD>
<HTML>
```

4 Vous allez maintenant saisir les différents éléments à l'intérieur de l'élément HEAD : TITLE et divers éléments META.

```
<HEAD>
<TITLE>"Mon site Web"</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=iso-8859-1">
<META name="author" content="votre nom">
<META name="version" content="2.7.1">
</HEAD>
```

Remarquez que ce contenu est identique à celui de l'élément HEAD de notre page d'accueil, à une exception près : comme nous apportons une modification majeure au site, le numéro de version devient 2.7.1.

5 Contrairement à une page Web « normale », un fichier d'ensemble de cadres est dépourvu d'élément BODY, remplacé par un élément FRAMESET. Vous y saisissez la définition de votre ensemble de cadres :

```
</HEAD>
   <FRAMESET rows="50, *">
      <FRAME src="barrenav.html" name="nav">
      <FRAME src="accueil.html" name ="princ">
   </FRAMESET>
</HTML>
```

Nos deux cadres empilés se comportent comme des lignes et sont donc définis dans l'ensemble de cadres comme tels (rows). Si nous avions choisi des cadres juxtaposés, donc en colonnes, nous aurions employé cols. Remarquez la spécification de la hauteur du premier cadre, 50 pixels. Le signe * employé pour le second signifie « occuper le reste de l'écran ». Ces points seront étudiés dans la suite de cette section.

6 Tous les agents utilisateurs ne sont pas en mesure d'afficher les cadres. Une bonne précaution consiste à prévoir une alternative, à l'aide d'un élément NOFRAMES. Si l'agent utilisateur ne peut afficher les cadres (ou est configuré pour ne pas le faire), il restitue le contenu de l'élément NOFRAMES.

```
    <FRAME src="accueil.html" name ="princ">
    <NOFRAMES>
      <P>Ce document a recours &agrave; un jeu d'encadrement
      qui contient :
        <UL>
          <LI><A href="barrenav.html">Une barre de
          navigation </A>
          <LI><A href="accueil.html">Une page
          d'accueil</A>
        </UL>
      </P>
    </NOFRAMES>
</FRAMESET>
</HTML>
```

7 Enregistrez votre fichier sous le nom *index.html*.



index.html ?

Vous pourriez vous demander pourquoi ce fichier est enregistré sous ce nom. Lorsque qu'il reçoit un URI dépourvu de nom de fichier, tout agent utilisateur recherche par défaut un fichier nommé *index.html*. Cela évite d'avoir à mémoriser le nom de la première page de votre site : seul l'URI vers son dossier est nécessaire, soit donc dans notre cas *…/MonSite*.

L'élément `FRAMESET` spécifie la disposition de la fenêtre principale de l'utilisateur selon des subdivisions rectangulaires : lignes et colonnes.

Nous avons employé ici l'attribut `rows`. Celui-ci spécifie la disposition des cadres horizontaux. Sa valeur est une liste de longueurs en pixels, en pourcentage ou relatives, séparées par des virgules. La valeur par défaut est `100 %`, c'est-à-dire une seule rangée.

Prenons un exemple :

```
<FRAMESET rows="20,25%,*">
...reste de la définition...
</FRAMESET>
```

Vous créez ici trois lignes. La première possède une hauteur fixe de 20 pixels (pour recevoir par exemple des icônes de navigation dont les dimensions sont connues). La seconde occupe 25 % de l'espace qui reste disponible et la troisième le solde (soit 75 %).

Nous aurions pu préférer une disposition en colonnes, à l'aide de l'attribut `cols`, qui spécifie la disposition des cadres verticaux. Sa

valeur est une liste de longueurs en pixels, en pourcentage ou relatives, séparées par des virgules. La valeur par défaut est `100 %`, c'est-à-dire une seule colonne.

Il est possible de combiner des attributs `rows` et `cols`. Si l'attribut `rows` est absent, chaque colonne occupe la hauteur entière de la page. Si l'attribut `cols` est absent, chaque rangée occupe la largeur entière de la page. Si aucun de ces attributs n'est présent, le cadre occupe la totalité de la page.

Les cadres sont créés de gauche à droite pour les colonnes et de haut en bas pour les lignes. Quand les deux attributs sont spécifiés, les cadres sont créés ligne par ligne du haut vers le bas, les colonnes étant placées successivement de gauche à droite dans la rangée supérieure, de gauche à droite dans la rangée suivante, etc.

Voici un exemple de grille 2×3 :

```
<FRAMESET rows="30%,70%" cols="33%,34%,33%">
...reste de la définition...
</FRAMESET>
```

Les longueurs absolues dont le total n'est pas égal à 100 % de l'espace disponible réel devraient être ajustées par l'agent utilisateur. Quand ce total est inférieur à 100 %, l'espace restant devrait être alloué proportionnellement à chaque vue. Quand il est supérieur, chaque vue devrait être réduite en fonction de la proportion de l'espace total qui lui est spécifiée.

Je pense que vous commencez à comprendre pourquoi il est fortement conseillé de tracer sur papier l'aspect voulu d'un jeu d'encadrement avant de commencer à le programmer…

7.2. Contenu d'un jeu d'encadrement : élément FRAME

Vous avez donc créé un jeu d'encadrement fort simple. Celui-ci est toutefois pour le moment totalement dépourvu de contenu. Vous devez avoir compris que cette page allait remplacer l'actuelle page d'accueil. Celle-ci est divisée en deux pages distinctes, affichées chacune dans un cadre propre : la partie navigation dans le cadre du haut, et le texte d'accueil dans le second cadre. Créez maintenant ces deux pages, à partir de l'ancienne page d'accueil.

1 Ouvrez dans le Bloc-Notes la dernière version de la page d’accueil, *pageacc_6_1.html*.

2 Vous allez d’abord créer la barre de navigation. Modifiez tout d’abord le contenu des éléments TITLE et META, comme suit :

```
<HEAD>
<TITLE>"Barre de navigation"</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
  charset=iso-8859-1">
<META name="author" content="votre nom">
<META name="version" content="2.7.1">
</HEAD>
```

3 Cette page doit être une simple barre de navigation. Remplacez la totalité du contenu actuel de l’élément BODY par ce qui suit (en faisant attention, vous pouvez vous éviter un peu de saisie…) :

```
<BODY>
<TABLE border="0" align="center">
   <TR>
      <TD><A href="accueil.html" target=princ>
        Accueil</A></TD>
      <TD><A href="region.html" target=princ>
        Ma r&eacute;gion</A></TD>
      <TD><A href="famille.html" target=princ>
        Ma famille</A></TD>
      <TD><A href="passions.html" target=princ>
        Mes passions</A></TD>
      <TD><A href="mailto:votre_nom@votre_FAI">
          <IMG src="Images/Boites009.gif" height="50"
          alt="Ecrivez-moi !"
           border="0"></A></TD
   </TR>
</TABLE>
</BODY>
```

Remarquez que nous avons éliminé les couleurs attribuées précédemment à la page, dans l’élément BODY : vous savez désormais définir vos propres couleurs et arrière-plans.

4 C’est tout pour la barre de navigation. Enregistrez le fichier sous le nom *barrenav.html*.

5 Rechargez dans le Bloc-Notes la page *pageacc_6_1.html*.

6 Vous allez cette fois créer la page d’accueil. Modifiez le contenu des éléments TITLE et META, comme suit :

```
<HEAD>
<TITLE>"Page d'accueil"</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
  charset=iso-8859-1">
```

```
<META name="author" content="votre nom">
<META name="version" content="2.7.1">
</HEAD>
```

7 Modifiez comme suit le contenu de l'élément `BODY`. Comme précédemment, vous devriez pouvoir vous éviter un peu de saisie… :

```
<BODY>
<H1 align="center">Ma page d'accueil</H1>
<HR>
<H2 align="center">Bienvenue sur <EM>mon</EM>
  site.</H2>
<DIV align="center">
<P>Vous trouverez sur ce site des
  <STRONG>informations</STRONG> :</P>
<UL>
<LI>sur ma r&eacute;gion</A> ;</LI>
<LI>sur ma famille</A> ;</LI>
<LI>sur mes passions</A>.</LI>
</UL>
<P>
Naviguez dans le site &agrave; l'aide de la barre de
navigation propos&eacute;e ci-dessus.
</P>
<HR>
<ADDRESS>
<A href="../ORignal/">Olivier Rignal</A> et
<A href="../JBogue/">Justin Bogue</A> sont les
personnes &agrave; contacter pour tout
<A href="probleme">probl&egrave;me</A>
  rencontr&eacute; sur le site.
</ADDRESS>
<!-- Des remarques l&eacute;gales ne seraient pas
     superflues -->
</DIV>
</BODY>
```

Remarquez le recours à un élément `DIV` pour obtenir le centrage du contenu de la page, après les titres.

8 Enregistrez le fichier sous le nom `accueil.html`.

Voici les listings complets des trois fichiers créés à ce stade :

Listing 7-1 : Index.html

```
<!DOCTYPE HTML PUBLIC
   "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
  <HEAD>
```

```
    <TITLE>"Mon site Web"</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
      charset=iso-8859-1">
    <META name="author" content="votre nom">
    <META name="version" content="2.7.1">
   </HEAD>
   <FRAMESET rows="50, *">
      <FRAME src="barrenav.html" name="nav">
      <FRAME src="accueil.html" name ="princ">
      <NOFRAMES>
        <P>Ce document a recours &agrave; un jeu
           d'encadrement qui contient :
           <UL>
             <LI><A href="barrenav.html">Une barre
               de navigation </A>
             <LI><A href="accueil.html">Une page
               d'accueil</A>
          </UL>
       </P>
     </NOFRAMES>
   </FRAMESET>
</HTML>
```

Listing 7-2 : barrenav.html

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>"Barre de navigation"</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
       charset=iso-8859-1">
    <META name="author" content="votre nom">
    <META name="version" content="2.7.1">
  </HEAD>
  <BODY>
    <TABLE border="0" align="center">
      <TR align="center">
        <TD><A href="accueil.html" target=princ>
          Accueil</A></TD>
        <TD><A href="region.html" target=princ>
          Ma r&eacute;gion</A></TD>
        <TD><A href="famille.html" target=princ>
          Ma famille</A></TD>
        <TD><A href="passions.html" target=princ>
          Mes passions</A></TD>
        <TD><A href="mailto:votre_nom@votre_FAI">
            <IMG src="Images/Boites009.gif"
             height="50" alt="Ecrivez-moi !"
             border="0"></A></TD
      </TR>
    </TABLE>
```

```
  </BODY>
</HTML>
```

Listing 7-3 : accueil.html

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>"Page d'accueil"</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
     charset=iso-8859-1">
    <META name="author" content="votre nom">
    <META name="version" content="2.7.1">
  </HEAD>
  <BODY>
    <H1 align="center">Ma page d'accueil</H1>
    <HR>
    <H2 align="center">Bienvenue sur <EM>mon</EM>
      site.</H2>
    <DIV align="center"
    <P>Vous trouverez sur ce site des
       <STRONG>informations</STRONG> :</P>
    <UL>
      <LI>sur ma r&eacute;gion</A> ;</LI>
      <LI>sur ma famille</A> ;</LI>
      <LI>sur mes passions</A>.</LI>
    </UL>
    <P>
Naviguez dans le site &agrave; l'aide de la barre
de navigation propos&eacute;e ci-dessus.
    </P>
    <HR>
    <ADDRESS>
      <A href="../ORignal/">Olivier Rignal</A> et
      <A href="../JBogue/">Justin Bogue</A> sont
      les personnes &agrave; contacter pour tout
      <A href="probleme">probl&egrave;me</A> rencontr&eacute;
      sur le site.
    </ADDRESS>
    <!-- Des remarques l&eacute;gales ne seraient pas
     superflues -->
    </DIV>
  </BODY>
</HTML>
```

Vous pouvez maintenant examiner votre travail dans votre navigateur. Ne testez pas encore les liens de la barre de navigation.



Figure 7.2 : *Aspect du nouveau jeu d'encadrement*

Quelques problèmes se posent dans la barre de navigation. Tout d'abord, en ayant éliminé les couleurs, l'icône retenue pour envoyer un courriel ressort de façon un peu désagréable. Vous pourriez définir votre propre charte graphique (et vous le ferez probablement), et savez où chercher une icône plus adaptée, grâce au Chapitre 6.

Plus gênant, bien que nous ayons pris la précaution d'éliminer les bordures du tableau et d'ajuster la hauteur de l'icône, celle-ci n'apparaît pas totalement. Une barre de défilement est donc proposée sur la droite de l'écran. En outre, la barre séparant les deux cadres n'est pas du meilleur goût. Remarquez d'ailleurs qu'il est possible de la cliquer-déposer pour modifier la taille du cadre de la barre de navigation, ce qui n'est pas ce que nous souhaitons. Il est possible de remédier à ces différents points.

1 Ouvrez à nouveau le fichier *index.html*.

2 Modifiez comme suit l'élément `FRAME` de la barre de navigation :

```
<FRAME src="barrenav.html" name="nav" noresize
  scrolling="no" frameborder="0" marginheight="0">
<FRAME src="accueil.html" name ="princ" frameborder="0">
```

Nous reviendrons sous peu plus en détail sur les attributs employés.

3 Enregistrez le fichier sans modifier son nom. Examinez-le à nouveau dans votre navigateur.



Figure 7.3 : *Aspect du jeu d'encadrement modifié*

L'aspect est désormais beaucoup plus sympathique. Avant de procéder à un test plus approfondi, il est nécessaire de veiller à ce que les cibles des liens de la barre de navigation soient valides. Si les fichiers *famille.html* et *passions.html* existent bien, ce n'est pas le cas du fichier *region.html*. Réglez ce problème maintenant.

4 Ouvrez le fichier *region1_6_3.html*, puis, sans le modifier, enregistrez-le sous le nom *region.html*, sans fermer le Bloc-Notes ni le fichier.

5 Testez maintenant les différents liens de la barre de navigation. Vous vous déplacez aisément d'une page à l'autre, sans avoir à revenir à la page d'accueil. Comme toutefois les liens sont dorénavant toujours présents à l'écran, ceux situés sur les pages secondaires deviennent superflus. Les éliminer améliorera l'aspect de ces pages, tout en facilitant leur maintenance ultérieure.



Figure 7.4 : *Supprimer les liens du bas de la page améliorera son aspect*

6 Revenez au fichier *region.html*, toujours ouvert. Modifiez son numéro de version comme suit :

```
<META name ="version" content="2.7.1">
```

7 Pour bien faire, supprimez les deux premiers éléments `OBJECT` (l’applet Python et l’animation MNG), puis supprimez les liens situés en bas de la page. Les lignes à supprimer sont les suivantes :

```
<!-- Essayer d'abord un applet en Python -->
<OBJECT title="Coucher de soleil sur le lac de
  Linciel" classid="Linciel.py">
  <!-- Sinon, essayer l'animation MNG -->
  <OBJECT data="Linciel.mng" type="application/mng">
...
<P align="center">
      <A href="pageacc1_6_1.html">Retour vers la page
      d'accueil</A>
    </P>
```

8 Vous pouvez modifier le commentaire avec l’élément `OBJECT` *jpg*. Enregistrez ensuite le fichier sous le même nom.

9 Ouvrez successivement les fichiers *famille.html* et *passions.html*. Modifiez les numéros de version en `2.7.1` et supprimez les liens, puis enregistrez-les sans modifier leur nom.

Désormais, vous passez d’une page à l’autre aisément, d’un simple clic.

Il est bien sûr possible à ce stade d’encore améliorer le site. À l’aide des informations obtenues lors de la lecture des chapitres précédents (essentiellement le Chapitre 6), vous pourriez notamment :

- Ajouter un arrière-plan aux deux cadres. En toute logique, un bandeau serait parfait pour la barre de navigation (puisque la taille est fixe), tandis qu’une mosaïque, éventuellement de type filigrane, serait parfaite pour le cadre principal.
- Ajouter un arrière-plan aux cellules du tableau de la barre de navigation, ou remplacer les liens texte par des dessins ou icônes employant des polices plus sophistiquées. Vous verrez dans le Chapitre 9, traitant des scripts, comment rendre cette barre encore plus attractive.

Vous pourriez également créer d'autres barres de navigation, adaptées à l'affichage de pages de niveau inférieur des pages déjà créées (par exemple, un détail des passions), voire modifier totalement le jeu d'encadrement en le remplaçant par un autre, comme vous allez le découvrir dans la suite de ce chapitre. Les possibilités ne dépendent que de votre imagination !

Voici le nouvel état du listing du fichier *index.html*.

Listing 7-4 : index.html

```
<!DOCTYPE HTML PUBLIC
   "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
  <HEAD>
    <TITLE>"Mon site Web"</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
      charset=iso-8859-1">
    <META name="author" content="votre nom">
    <META name="version" content="2.7.2">
   </HEAD>
   <FRAMESET>
     <FRAME src="barrenav.html" name="nav" noresize
        scrolling="no"
        frameborder="0" marginheight="0">
     <FRAME src="accueil.html" name ="princ"
        frameborder="0">
     <NOFRAMES>
        <P>Ce document a recours &agrave; un jeu
           d'encadrement qui contient :
           <UL>
             <LI><A href="barrenav.html">
               Une barre de navigation </A>
             <LI><A href="accueil.html">
               Une page d'accueil</A>
          </UL>
       </P>
     </NOFRAMES>
   </FRAMESET>
</HTML>
```

Il est toutefois temps de comprendre comment s'effectue toute cette magie, en examinant les différents constituants de l'élément FRAMESET. Commençons par les attributs de l'élément FRAME.

Attributs essentiels de l'élément FRAME : name et src

L'élément FRAME définit le contenu et l'apparence d'un unique cadre.

L'attribut name assigne un nom au cadre courant. Ce nom est employé comme cible des liens : le cadre dans lequel ouvrir un document. Bien que facultatif en théorie, cet attribut est en pratique presque toujours nécessaire.

```
<FRAME src="barrenav.html" name="nav" noresize
    scrolling="no"
    frameborder="0" marginheight="0">
<FRAME src="accueil.html" name ="princ"
    frameborder="0">
```

Un nom de cadre doit répondre à certains impératifs. Il doit commencer avec un caractère alphanumérique, pas par un symbole. Il existe toutefois un certain nombre de « noms réservés » qui possèdent des propriétés spéciales. Ces noms commencent par un trait de soulignement. Il s'agit de _blank, _self, _parent, et _top. Chacun possède un but spécial dans les cadres, et accomplit quelque chose d'unique. Ces noms réservés et leur emploi seront examinés un peu plus loin dans ce chapitre.

L'attribut src spécifie la localisation du contenu initial à placer dans le cadre défini par l'élément FRAME :

```
<FRAME src="barrenav.html" name="nav" noresize
    scrolling="no"
    frameborder="0" marginheight="0">
<FRAME src="accueil.html" name ="princ"
    frameborder="0">
```



Contenu et définition des cadres

La définition d'un jeu d'encadrement, telle que spécifiée par les valeurs des attributs src des éléments FRAME, ne change jamais. En revanche, le contenu de l'un ou de plusieurs de ses cadres peut changer. Après une telle modification, la définition du jeu d'encadrement ne reflète plus la situation courante de ses cadres.

Attributs de mise en forme de l'élément FRAME

Ces attributs conditionnent la façon dont les agents utilisateurs restituent le cadre et influent sur son comportement.

L'attribut booléen `noresize` indique à l'agent utilisateur que le cadre ne doit pas être redimensionnable : s'il est présent, il est impossible d'effectuer un glisser-déposer pour modifier la taille du cadre.

```
<FRAME src="barrenav.html" name="nav" noresize
    scrolling="no"
    frameborder="0" marginheight="0">
```

L'attribut `scrolling` autorise ou interdit la présence de barres de défilement pour la fenêtre du cadre. Les valeurs possibles sont `auto` (une barre de défilement est présente si nécessaire. C'est la valeur par défaut), `yes` (mécanisme de défilement toujours présent) et `no` (mécanisme de défilement toujours absent).

```
<FRAME src="barrenav.html" name="nav" noresize
    scrolling="no"
    frameborder="0" marginheight="0">
```

L'attribut `frameborder` procure à l'agent utilisateur une indication sur la bordure du cadre. Les valeurs possibles sont `1` (présence d'une séparation entre ce cadre et chacun des cadres adjacents. C'est la valeur par défaut) et `0` (pas de séparation entre ce cadre et chacun des cadres adjacents).

Remarquez que des séparateurs peuvent néanmoins être dessinés à côté de ce cadre, s'ils sont spécifiés par les cadres adjacents. Si vous ne voulez pas de séparation entre plusieurs cadres, spécifiez `frameborder="0"` pour tous les cadres concernés, comme nous l'avons fait sur notre site :

```
<FRAME src="barrenav.html" name="nav" noresize
    scrolling="no"
    frameborder="0" marginheight="0">
<FRAME src="accueil.html" name ="princ"
    frameborder="0">
```

Les attributs `marginwidth` et `marginheight` spécifient respectivement la quantité d'espace à laisser entre le contenu du cadre et ses marges latérales (gauche et droite) et haute et basse. La valeur, exprimée en pixels, doit être supérieure à zéro. Les valeurs par défaut dépendent de l'agent utilisateur. Comme les principaux navigateurs présentent sur ce point des différences sensibles, mieux vaut spécifier

explicitement des valeurs, surtout pour les barres de navigation et les cadres à dimensions fixes.

```
<FRAME src="barrenav.html" name="nav" noresize
    scrolling="no"
    frameborder="0" marginheight="0">
```

Autres attributs de l'élément FRAME

Comme pour la plupart des autres éléments, un élément `FRAME` peut posséder des attributs `id`, `class` (identifiants internes au document), `title` (titre de l'élément) et `style` (informations de style en ligne), ainsi qu'un attribut `longdesc` qui spécifie un lien vers une description longue du cadre. Cette description devrait suppléer à la brève description fournie à l'aide de l'attribut `title`. Elle peut être particulièrement utile aux agents utilisateurs non visuels.

L'attribut `longdesc` permet de rendre les documents avec des cadres plus accessibles aux personnes utilisant des agents utilisateurs non visuels. Remarquez toutefois que les descriptions longues associées aux cadres sont attachées à ceux-ci et non à leur contenu. Ce dernier pouvant varier au fil du temps, la description longue initiale peut devenir inadéquate pour les contenus ultérieurs du cadre. Mieux vaut notamment ne pas inclure une image comme seul contenu d'un cadre.

Le document avec jeu d'encadrement suivant décrit deux cadres. Le cadre de gauche contient une table des matières et celui de droite contient initialement l'image d'une autruche :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
<HEAD>
<TITLE>Un document avec jeu d'encadrement mal fait</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
   <FRAME src="table_des_matieres.html">
   <FRAME src="autruche.gif" longdesc="autruche-desc.html">
</FRAMESET>
</HTML>
```

Remarquez que l'image est incluse dans le cadre indépendamment de tout élément HTML : l'attribut `longdesc` constitue le seul moyen de spécifier un texte de remplacement. Si le contenu du cadre de droite change (par exemple, l'utilisateur choisit dans la table des matières un serpent à sonnette), les utilisateurs n'auront aucun accès textuel au nouveau contenu du cadre.

Il ne faut donc pas placer directement une image dans un cadre comme valeur de l'attribut `src`. L'image doit plutôt être spécifiée dans un document HTML séparé, dans lequel elle est annotée à l'aide du texte de remplacement adéquat :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
<HEAD>
<TITLE>Un document avec jeu d'encadrement bien fait</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
   <FRAME src="table_des_matieres.html">
   <FRAME src="autruche-conteneur.html">
</FRAMESET>
</HTML>
```

```
<!-- Dans le document autruche-conteneur.html: -->
<HTML>
<HEAD>
<TITLE>L'autruche rapide et puissante</TITLE>
</HEAD>
<P>
<OBJECT data="autruche.gif" type="image/gif">
Ces autruches ont vraiment bon go&ucirc;t !
</OBJECT>
</HTML>
```

Enfin, un attribut `target` peut être placé dans un élément `FRAME`, définissant ainsi une cible par défaut pour tous les liens de ce cadre. L'attribut `target` est étudié plus en détail dans la section suivante.



Définition et contenu

Un document qui contient une définition de cadre ne doit pas comporter le contenu de ce cadre. L'exemple suivant est illégal, puisque le contenu du deuxième cadre se trouve dans le même document que le jeu d'encadrement.

```
<FRAMESET cols="50%,50%">
  <FRAME src="contenu_cadre1.html">
  <FRAME src="#ancre_du_meme_document">
  <NOFRAMES>
  ...un texte...
  <H2><A name="ancre_du_meme_document">Passage important</A>
  </H2>
  ...un texte...
  </NOFRAMES>
</FRAMESET>
```

7.3. Détermination du cadre cible : attribut target

Une fois le jeu d'encadrement créé, vous devez disposer d'un moyen de naviguer entre les cadres. La plupart des jeux d'encadrement consistent en un cadre principal entouré d'autres cadres, comme des bannières, des menus de navigation ou d'autres éléments.

Que se passe-t-il toutefois lorsque vous cliquez sur un lien situé à l'intérieur d'un cadre ? D'habitude, cela ouvre une nouvelle page à l'intérieur de la fenêtre active, donc du même cadre. Ceci peut rendre les choses difficiles à lire lors de l'utilisation d'un menu de navigation étroit ! Heureusement, lorsque vous cliquez sur un lien dans un ensemble de cadres, vous pouvez spécifier que la nouvelle page doit s'ouvrir :

- Dans le même cadre.
- Dans un autre cadre du jeu d'encadrement.
- Dans un autre cadre à l'intérieur d'un cadre enfant ou parent (dans le cas de cadres imbriqués, abordés plus loin).
- Dans un nouveau jeu d'encadrement qui remplace le jeu existant et toutes les fenêtres qui pouvaient s'y trouver.
- Dans un autre cadre, forçant un autre cadre à se relancer ou à ouvrir une nouvelle page.
- Dans une nouvelle fenêtre du navigateur.
- Dans la fenêtre actuelle en remplaçant l'actuel jeu d'encadrement.

Comme vous pouvez le voir, il existe de nombreuses façons de manipuler le contenu des cadres à l'aide de liens. Par défaut, les liens s'ouvrent dans le même cadre.

Pour ouvrir un lien dans un autre cadre, vous employez l'attribut `target` :

```
<A href="exemple.html" target="cible">
```

Si toutefois vous souhaitez ouvrir un cadre à l'intérieur d'un autre cadre du jeu d'encadrement, vous devez d'abord préciser le nom de chaque cadre, et cibler ce cadre. Par exemple :

```
<FRAME name="contenu">
```

nomme un cadre précis nommé `contenu`. Un clic sur le lien

```
<A href="info.html" target="contenu">
```

ouvrira le document Web nommé *info.html* à l'intérieur du cadre nommé `contenu`.

L'attribut `target` peut se placer sur les éléments qui créent des liens (`A`, `LINK`), des images cliquables (`AREA`) et des formulaires (`FORM`).

Vous pouvez aussi cibler tous les cadres d'un ensemble, et tous les liens d'un cadre, en plaçant l'attribut `target` dans l'élément `FRAME`.

```
<FRAME name="barrenav" target="contenu">
```

ouvre un cadre nommé `barrenav`, qui ouvrira par défaut tous ses liens dans le cadre nommé `contenu`. Comme vous pouvez le voir, les possibilités sont immenses.

Noms réservés

Nous avons précédemment évoqué l'existence de noms réservés, qui débutent par un trait de soulignement : `_blank`, `_self`, `_parent` et `_top`.

Ces noms réservés **devraient** imposer à l'agent utilisateur de fonctionner comme suit :

- Un lien qui cible `_blank` ouvre une page dans une nouvelle fenêtre dépourvue de nom.
  Utilisez `target=_blank` lorsque votre lien renvoie en dehors de votre site. Par exemple, si vous proposez une liste de liens, ceux qui appartiennent à d'autres personnes doivent s'ouvrir dans de nouvelles fenêtres. En effet, votre site n'est probablement pas le seul à utiliser les cadres : rien n'est plus énervant qu'un site en cadres à l'intérieur du petit cadre de contenu d'un autre site en cadres. En outre, vous ne voudriez peut-être pas être associé au propriétaire du contenu de ce site : si le lien s'ouvre dans votre ensemble, il apparaîtra comme faisant partie de votre site, ce qui peut être très désagréable.



Anciennes versions

Avec les anciennes versions de Netscape Navigator, `_new` possède un effet analogue à `_blank`, mais ne fait pas partie du standard HTML.

- La cible _self ouvre une page à l'intérieur du cadre actif. C'est le comportement par défaut de beaucoup d'agents utilisateurs : en l'absence de cible spécifiée, le document nommé par le lien s'ouvre dans le cadre actif. Ceci est utile pour les liens situés à l'intérieur du cadre principal, généralement censés ouvrir des contenus différents dans ce cadre.
- Le nom réservé _parent s'applique aux cadres imbriqués. Lorsque vous ciblez le cadre _parent, vous ciblez réellement le cadre dans lequel réside votre cadre actuel.
- Enfin, _top ouvre le lien dans la fenêtre en cours, remplaçant la totalité du jeu d'encadrement actif. Une fenêtre de navigation peut être considérée comme un conteneur. Celui-ci peut renfermer un jeu d'encadrement, à l'intérieur duquel peut se trouver un autre jeu d'encadrement. Lorsque vous lancez un lien dans un autre cadre, le contenu est mélangé dans le conteneur. En l'ouvrant en revanche dans _top, le contenu du conteneur est remplacé en totalité par la ou les pages auxquelles mène ce lien. Vous emploierez par exemple _top pour ouvrir un nouveau fichier de définition de jeu d'encadrement, différent du jeu actif.

Remarquez bien l'emploi de « *devraient* »… En pratique, quelques précautions complémentaires sont nécessaires lors de l'emploi de jeux d'encadrement imbriqués, comme vous le verrez plus loin. Les agents utilisateurs, dont les principaux navigateurs, peuvent interpréter différemment selon les situations ces noms réservés, notamment _parent.



Conflits de noms de cadres

À un moment donné, il peut arriver que deux cadres possèdent le même nom. C'est souvent le résultat d'une ouverture d'un ensemble de cadres dans un ensemble existant. HTML 4.01 dispose de règles spéciales pour la gestion de ce genre de conflits. Le navigateur recherche d'abord le cadre nommé dans l'ensemble le plus profond. Si aucun cadre de ce nom n'existe, il cherche dans le cadre parent, puis dans l'ensemble parent, et ainsi de suite. S'il ne parvient toutefois pas à identifier de cadre portant ce nom, il ouvre un nouveau cadre et lui attribue ce nom.

Établissement de la cible par défaut des liens

Quand, dans une même document, plusieurs liens désignent la même cible, il est possible de spécifier la cible une seule fois et de se passer de l'attribut `target` de chaque élément. Pour ce faire, définissez dans le document l'attribut `target` sur un élément `BASE`.

Par exemple, pour spécifier que tous les liens situés dans le document *passions.html* doivent ouvrir un nouveau fichier dans le cadre nommé `affichage`, vous définissez comme suit un élément `BASE` :

```
...
<META name="version" content="2.7.1">
<BASE href="MonSite" target="affichage">
</HEAD>
...
```

Sémantique de cible

La détermination par l'agent utilisateur du cadre cible n'est pas toujours évidente. HTML 4.01 recommande que les agents utilisateurs déterminent le cadre cible dans lequel charger une ressource reliée selon la préséance suivante (de la priorité la plus élevée à la plus basse) :

- Si l'attribut `target` d'un élément vise un cadre connu, quand l'élément est activé (un clic sur un lien ou la soumission d'un formulaire), la ressource désignée par l'élément devrait se charger dans le cadre cible.
- Si cet élément est dépourvu d'attribut `target` et qu'en revanche l'élément `BASE` en possède un, c'est l'attribut `target` de l'élément `BASE` qui détermine le cadre.
- Si ni cet élément ni l'élément `BASE` ne se réfèrent à une cible, la ressource désignée par l'élément devrait se charger dans le cadre qui contient l'élément (comportement par défaut identique à une cible `_self`).
- Si un attribut `target` se réfère à un cadre `C` inconnu, l'agent utilisateur devrait créer une nouvelle fenêtre et un nouveau cadre, puis assigner le nom `C` au cadre et, enfin, charger la ressource désignée par l'élément dans le nouveau cadre.

Les agents utilisateurs devraient fournir aux utilisateurs un mécanisme permettant de surclasser l'attribut `target`.

Remarquez qu'une fois de plus les comportements par défaut préconisés par HTML pour les agents utilisateurs sont d'une efficacité certaine : vous pouvez très souvent vous abstenir de spécifier différents attributs en vous reposant sur le comportement par défaut. C'est toutefois une attitude un peu risquée, car un agent utilisateur au comportement inhabituel peut amener des résultats catastrophiques : par sécurité, mieux vaut spécifier tout ce qui peut l'être. Cela est particulièrement valable pour l'attribut `target`.

7.4. Ensembles de cadres imbriqués

Les jeux d'encadrement peuvent s'imbriquer à n'importe quel niveau.

Dans l'exemple suivant, l'élément `FRAMESET` externe divise l'espace disponible en trois colonnes presque égales. L'élément `FRAMESET` interne partage alors la deuxième colonne en deux rangées de hauteur inégale.

```
...
<FRAMESET cols="33%, 33%, 34%">
     <FRAME name="cadre1" src="cadre1.html">
     <FRAMESET rows="40%, 60%">
        <FRAME name="cadre211" src="cadre211.html">
        <FRAME name="cadre212" src="cadre212.html">
</FRAMESET>
     <FRAME name="cadre3" src="cadre3.html">
</FRAMESET>
...
```

Le schéma de l'aspect de ce jeu d'encadrement est présenté dans la figure suivante.

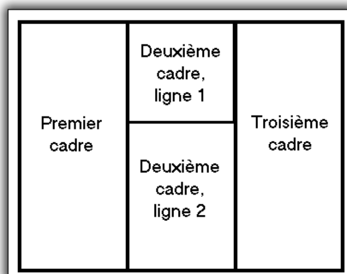


Figure 7.5 :
Jeux d'encadrement imbriqués

Lorsque des jeux d’encadrement sont imbriqués, certains cadres sont les « parents » des autres. Cela est sans importance la plupart du temps, mais devient capital lorsque vous devez gérer les scripts et que vous utilisez les noms réservés, avec comme cible `_parent`.

Revenez à l’exemple présenté ci-dessus. Ce jeu crée un ensemble de quatre cadres. Le jeu principal est divisé en trois colonnes, la colonne centrale étant divisée en deux lignes. Considérez le lien suivant :

```
<A href="test_parent.html" target="_parent">Lien</A>
```

Nous avons placé ce lien dans les fichiers nommés *cadre1.html* et *cadre2l1.html*. Le fichier *cadre2l2.html* contient en revanche le lien :

```
<A href="test_parent.html" target="_self">Lien</A>
```

Enfin, le fichier *cadre3.html* contient le lien :

```
<A href="test_parent.html" target="_top">Lien</A>
```

En ouvrant le fichier *cadresimbriques.html* dans votre navigateur, vous voyez ce qui est présenté dans la figure suivante.

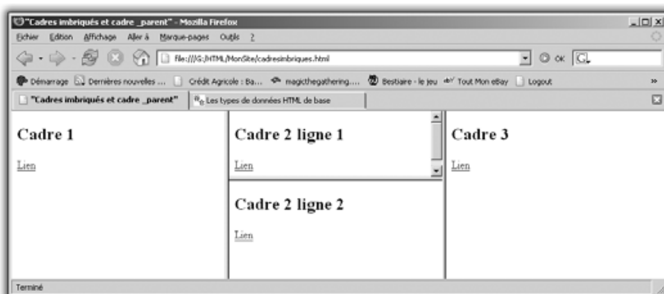


Figure 7.6 : *Jeux d’encadrement imbriqués*

Un clic sur le lien du cadre nommé `cadre1` ouvre le fichier *test_parent.html* dans la totalité de la page.



Figure 7.7 : *Le _parent de cadre1 est la totalité de la fenêtre*

Un clic sur le lien du cadre de la seconde ligne de la seconde colonne ouvre le fichier *test_parent.html* dans ce cadre.

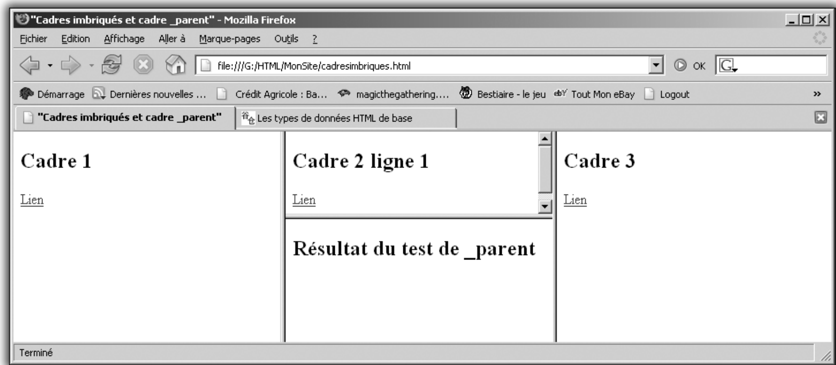


Figure 7.8 : *Le _self de cadre2 est bien lui-même*

Enfin, un clic sur le lien du cadre 3 ouvre aussi le fichier dans la totalité de la page : pour `cadre1` et `cadre2`, `_top` et `_parent` possèdent la même signification. Jusque-là, tout va bien.

En revanche, un clic sur le lien du cadre de la première ligne de la seconde colonne devrait ouvrir le fichier *test_parent.html* dans la totalité de cette colonne, remplaçant ce deuxième jeu d'encadrement. Malheureusement, tel n'est pas le cas, et le fichier s'ouvre comme si `_top` avait été employé, ou comme si vous aviez cliqué sur le lien de `cadre1` : c'est-à-dire sur un lien situé dans un cadre qui n'appartient pas lui-même à un jeu d'encadrement secondaire. Le résultat est identique à ce qui est présenté dans la figure de la page 276.

Que se passe-t-il ? Il semble en pratique que les agents utilisateurs considèrent souvent que des jeux d'encadrement imbriqués définis dans un unique fichier ne constituent qu'un unique jeu d'encadrement. Pour remédier à ce défaut, nous avons modifié comme suit le fichier de spécification du jeu d'encadrement principal, enregistré sous le nom *cadresimbriques2.html* :

```
...
<FRAMESET cols="33%, 33%, 34%">
   <FRAME name="cadre1" src="cadre1.html">
   <FRAME name="cadre2" src="cadre2.html">
   <FRAME name="cadre3" src="cadre3.html">
</FRAMESET>
...
```

Vous spécifiez donc ici un jeu d’encadrement unique. Remarquez l’apparition du fichier *cadre2.html*. Celui-ci est une nouvelle spécification de jeu d’encadrement :

```
...
<FRAMESET rows="40%, 60%">
   <FRAME name="cadre2l1" src="cadre2l1.html">
   <FRAME name="cadre2l2" src="cadre2l2.html">
</FRAMESET>
...
```

Nous possédons déjà les autres fichiers. À l’examen du fichier *cadresimbriques2.html* dans un navigateur, aucune différence avec l’aspect précédent n’apparaît, si ce n’est le titre et l’URL de la page.

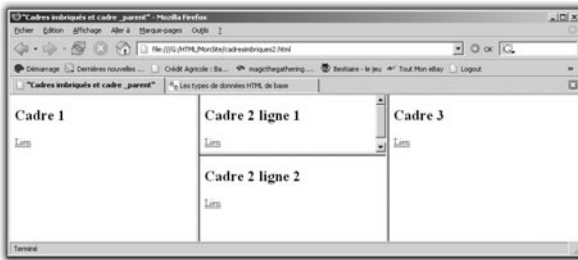


Figure 7.9 : *Jeux d’encadrement imbriqués*

Comme précédemment, un clic sur le lien du cadre nommé `cadre1` ouvre le fichier *test_parent.html* dans la totalité de la page. Un clic sur le lien du cadre de la seconde ligne de la seconde colonne ouvre le fichier *test_parent.html* dans ce cadre. Enfin, un clic sur le lien du cadre 3 ouvre aussi le fichier dans la totalité de la page. Aucune différence.

Cette fois cependant, un clic sur le lien du cadre de la première ligne de la seconde colonne ouvre bien le fichier *test_parent.html* dans la totalité de cette colonne, remplaçant ce deuxième jeu d’encadrement.



Figure 7.10 : *Le _parent de cadre2l1 est cette fois cadre2*

Lors de la réalisation de cadres imbriqués, mieux vaut donc séparer les spécifications des jeux concernés pour éviter tout problème de comportement du nom réservé `_parent`, ou vous en tenir à des cadres nommés explicitement désignés.

7.5. Partage de données entre cadres

Il est possible de partager des données entre plusieurs cadres en incluant celles-ci au moyen de l'élément `OBJECT`. L'élément `OBJECT` doit être placé dans l'élément `HEAD` du document de définition du jeu d'encadrement et nommé à l'aide d'un attribut `id`. Tout document qui est le contenu d'un cadre dans le jeu d'encadrement peut alors se référer à cet identifiant.

L'exemple suivant illustre la manière dont un script pourrait appeler un élément `OBJECT` défini pour le jeu d'encadrement entier (nous étudierons plus en détail les scripts dans le Chapitre 10).

Fichier de définition du jeu d'encadrement :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
  <HEAD>
    <TITLE>Voici un jeu d'encadrement avec un OBJECT dans HEAD
    </TITLE>
    <!-- Cet OBJECT n'est pas restitu&eacute ! -->
    <OBJECT id="mon_objet" data="data.bar"></OBJECT>
  </HEAD>
  <FRAMESET>
    <FRAME src="martine.html" name="martine">
  </FRAMESET>
</HTML>
```

Listing 7-5 : Fichier martine.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>La page de Martine</TITLE>
  </HEAD>
  <BODY>
  ...le d&eacute;but du document...
    <P>
    <SCRIPT type="text/javascript">
    parent.mon_objet.propri&eacute;t&eacute;
    </SCRIPT>
```

```
  ...le reste du document...
  </BODY>
</HTML>
```

D'autres cadres du jeu d'encadrement pourraient également faire référence à l'élément OBJECT.

7.6. Contenu de remplacement : élément NOFRAMES

Tous les agents utilisateurs ne sont pas en mesure de reconnaître les cadres. En outre, certains peuvent être configurés pour ne pas les afficher. Il est donc important de fournir un contenu de remplacement, ce qui est obtenu à l'aide de l'élément NOFRAMES.

L'élément NOFRAMES spécifie le contenu qui ne devrait être affiché que par les agents utilisateurs qui ne reconnaissent pas les cadres ou sont configurés pour ne pas les afficher. Les agents utilisateurs qui reconnaissent les cadres ne doivent afficher le contenu d'une déclaration NOFRAMES que s'ils sont configurés pour ne pas afficher les cadres. Ceux qui ne reconnaissent pas les cadres doivent afficher dans tous les cas le contenu de l'élément NOFRAMES.

L'élément NOFRAMES fait partie à la fois des DTD transitoires et des jeux d'encadrement. Dans un document qui fait appel au DTD de jeu d'encadrement, l'élément NOFRAMES est placé à la fin de la section FRAMESET du document.

C'est ce que nous avons fait dans notre fichier de définition de jeu d'encadrement *index.html* :

```
...
<FRAMESET rows="50, *">
  <FRAME src="barrenav.html" name="nav" noresize
     crolling="no" frameborder="0" marginheight="0">
  <FRAME src="accueil.html" name ="princ" frameborder="0">
  <NOFRAMES>
     <P>Ce document a recours &agrave; un jeu d'encadrement
        qui contient :
       <UL>
         <LI><A href="barrenav.html">Une barre de
           navigation </A>
         <LI><A href="accueil.html">Une page
           d'accueil</A>
       </UL>
```

```
    </P>
  </NOFRAMES>
</FRAMESET>
...
```

L'élément NOFRAMES peut être employé dans un document qui constitue la source d'un cadre et qui utilise le DTD transitoire. Cela permet aux auteurs d'expliquer l'objet du document, au cas où celui-ci serait vu indépendamment du jeu d'encadrement ou encore avec un agent utilisateur ne reconnaissant pas les cadres.

7.7. Cadres en ligne : élément IFRAME

Depuis sa version 3.0, Internet Explorer permettait de définir des cadres en ligne, qui « flottaient » dans la page. Ce dispositif a été repris dans HTML 4.01.

L'élément IFRAME permet d'insérer un cadre dans un bloc de texte. L'insertion d'un cadre en ligne dans un passage textuel revient un peu à y insérer un objet *via* l'élément OBJECT : ces éléments permettent tous deux l'insertion d'un document HTML au sein d'un autre, ils peuvent tous deux être alignés sur le texte environnant, etc.

Cet élément possède les mêmes attributs que l'élément FRAME, à l'exception de l'attribut noresize superflu puisqu'un cadre en ligne ne peut **jamais** être redimensionné.

Les informations qui doivent être insérées en ligne sont désignées par l'attribut src. En revanche, comme avec l'élément OBJECT, le contenu de l'élément IFRAME ne doit être affiché que par les agents utilisateurs qui ne reconnaissent pas les cadres ou qui sont configurés pour ne pas les afficher.

Voici un exemple de mise en œuvre d'un cadre flottant.

Listing 7-6 : cadreflottant.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
  <HEAD>
    <TITLE>Exemple de cadre flottant </TITLE>
  </HEAD>
  <BODY>
    <P>Ceci est un exemple de cadre flottant
```

```
      <IFRAME src="cadreflottant2.html"
             width="400" height="500"
             scrolling="auto" frameborder="1">
  [Votre agent utilisateur ne reconnaît pas les cadres ou
  n'est pas configur&eacute; pour les afficher pour l'instant.
  Cependant, vous pouvez visiter le
  <A href="cadreflottant2.html">document concerné.</A>]
      </IFRAME>
     plac&eacute; &agrave; l'int&eacute;rieur d'un texte.
     </P>
  </BODY>
</HTML>
```

Listing 7-7 : cadreflottant2.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>Cadre flottant</TITLE>
  </HEAD>
  <BODY>
  <H1>Quelques informations importantes</H1>
  <P>Les informations contenues dans ce document sont
affich&eacute;es dans un cadre flottant si l'agent
utilisateur reconna&icirc;t les cadres ou peuvent &ecirc;tre
affich&eacute;es dans une page propre si tel n'est pas le cas.
</P>
  </BODY>
</HTML>
```

Pour les agents utilisateurs qui reconnaissent les cadres, l'exemple suivant placera un cadre en ligne, entouré par une bordure, au milieu du texte.



Figure 7.11 : *Cadre flottant affiché par un navigateur reconnaissant les cadres*

En revanche, si l'agent utilisateur ne reconnaît pas les cadres ou est configuré pour ne pas les afficher, le fichier *cadreflottant.html* sera affiché.

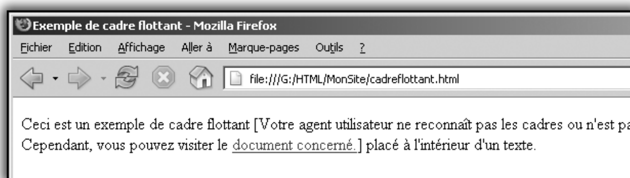


Figure 7.12 : *Cadre flottant affiché par un navigateur qui ne reconnaît pas les cadres*

Un clic sur le lien proposé à cet effet affiche le contenu du cadre flottant dans une fenêtre propre.



Figure 7.13 : *Le contenu du cadre flottant (celui du fichier cadreflottant2.html) est affiché dans une fenêtre propre*



Élément OBJECT

Vous pouvez également incorporer un document HTML dans un autre document HTML avec l'élément `OBJECT`, déjà abordé au Chapitre 6 ainsi que dans la section précédente, et sur lequel nous reviendrons encore.

7.8. Travail avec les jeux d'encadrement

Lorsque vous concevez un jeu d'encadrement, vous devez vous souvenir que son aspect à l'écran peut différer de ce que vous aviez en tête, et ce pour de multiples raisons. Par exemple, chaque barre de défilement présente sur un cadre réduit l'espace disponible à l'écran, ce qui rend le site de plus en plus exiguë et inconfortable. Pour améliorer le confort, réduisez le nombre de cadres.

Commencez par bien définir les raisons pour lesquelles vous souhaitez employer un cadre dans vos pages HTML. Désirez-vous faire de la place pour une bannière publicitaire ? Inclure une barre de navigation

sur un côté de l'écran (ou en haut ou en bas) ? Comparer deux documents côte à côte ? Mieux vaut toujours tracer par écrit un schéma de la structure de jeu d'encadrement que vous souhaitez concevoir, en y inscrivant les noms attribués aux cadres.

Les principales questions auxquelles vous devez impérativement répondre préalablement à la conception d'un jeu d'encadrement sont les suivantes :

- Quel cadre sera susceptible de subir les modifications les plus fréquentes ou les plus nombreuses ?
- Comment les utilisateurs navigueront-ils dans ce jeu d'encadrement ?
- En quoi ce jeu d'encadrement aidera-t-il votre public à apprécier le site ?

D'une certaine façon, les problèmes de conception d'un jeu d'encadrement sont très proches de ceux rencontrés lors de la conception de tableaux.

Un des principaux problèmes posés par les cadres, comme avec les tableaux, concerne la taille de l'écran. Si toutefois, dans une conception HTML typique, vous ne vous préoccupez généralement que de la largeur de l'écran lors de la réalisation de grands tableaux de présentation, les cadres occupent la totalité de la fenêtre du navigateur.

La plupart des utilisateurs naviguent avec une taille de fenêtre maximale : celle-ci occupe tout l'écran disponible. Ceci signifie que les résolutions type des écrans ont un rôle à jouer. Les résolutions les plus fréquentes sont désormais par ordre décroissant : 1024 × 768 (56 %), 1280 × 1024 (16 %) et 800 × 600 (12 %) et, les autres résolutions (1280 × 800 et 1152 × 864) ne représentant que moins de 8 % du parc. Il est donc raisonnable de considérer que l'écran de l'utilisateur possède une dimension au moins égale à 800 pixels de large pour 600 pixels de haut : tenez-en compte lors de votre conception, par exemple pour la taille d'une barre de navigation.

N'oubliez en outre pas que les cadres sont officiellement apparus dans HTML 4.0. Des agents utilisateurs plus anciens ne pourront peut être pas reconnaître un ou plusieurs des attributs décrits dans ce chapitre. Par exemple, « cibler un cadre » ou les attributs de largeur et de délimitation peuvent ne pas fonctionner correctement avec des navigateurs anciens. Pire encore, ils peuvent fonctionner d'une façon autre que celle

attendue. Même si la grande majorité des navigateurs prennent en charge depuis un temps certain la quasi-totalité des attributs abordés dans ce chapitre (comme Microsoft Internet Explorer et Netscape Navigator depuis leur version 3.0), mieux vaut toujours tester vos pages avec d'autres navigateurs avant de lancer votre site.

Sachez enfin que les jeux d'encadrement peuvent rendre la navigation d'avant en arrière, par le biais de l'historique de l'agent utilisateur, plus compliquée pour les utilisateurs.

7.9. Résumé

- Les cadres ou *frames* sont largement employés en HTML. Ils constituent le principal moyen d'afficher plusieurs HTML sur le même écran au même moment.
- Un fichier de définition de jeu d'encadrement possède une syntaxe analogue à celle d'un document HTML ordinaire, mais son code diffère. Ils utilisent un élément `DOCTYPE` différent en tête de page, et sont dépourvus d'élément `BODY`, remplacé par un élément `FRAMESET`.
- L'élément `FRAMESET` spécifie la disposition des vues dans la fenêtre principale de l'agent utilisateur.
- La spécification de l'attribut `rows` de l'élément `FRAMESET` définit le nombre de subdivisions horizontales dans un jeu d'encadrement. La spécification de l'attribut `cols` définit le nombre de subdivisions verticales. Les deux attributs peuvent être utilisés simultanément pour créer une grille.
- Au sein de l'élément `FRAMESET`, les différents cadres sont définis à l'aide d'éléments `FRAME`. Un élément `FRAME` définit le contenu et l'apparence d'un unique cadre.
- L'attribut `name` d'un élément `FRAME` assigne un nom au cadre concerné. Ce nom est employé comme cible des liens : le cadre dans lequel ouvrir un document. Un nom de cadre doit impérativement commencer par un caractère alphanumérique. Il existe toutefois un certain nombre de « noms réservés » qui possèdent des propriétés spéciales et débutent par un trait de soulignement (`_blank`, `_self`, `_parent`, et `_top`).
- L'attribut `src` d'un élément `FRAME` spécifie la localisation du contenu initial à placer dans le cadre défini par l'élément `FRAME`.

- Vous pouvez spécifier le mode de restitution d'un cadre par l'agent utilisateur à l'aide de différents attributs. `noresize` indique à l'agent utilisateur que le cadre ne doit pas être redimensionnable, l'attribut `scrolling` autorise ou interdit la présence de barres de défilement, `frameborder` spécifie la taille (ou l'absence) de la bordure du cadre. Les attributs `marginwidth` et `marginheight` spécifient respectivement la quantité d'espace à laisser entre le contenu du cadre et ses marges latérales (gauche et droite) et haute et basse.
- Par défaut, tout lien présent dans un cadre ouvre sa référence dans le même cadre. Pour ouvrir un lien dans un autre cadre, vous devez utiliser l'attribut `target`, suivi du nom du cadre cible. Vous pouvez définir un cadre cible par défaut pour les liens du cadre en employant l'attribut `target` dans un élément `BASE`.
- HTML 4.01 recommande des règles de priorités que devraient respecter les agents utilisateurs pour déterminer le cadre cible dans lequel charger une ressource liée.
- Il est possible d'imbriquer un nombre quelconque de jeux d'encadrement.
- Il est possible de partager des données entre plusieurs cadres en incluant celles-ci au moyen de l'élément `OBJECT`. L'élément `OBJECT` doit être placé dans l'élément `HEAD` du document de définition du jeu d'encadrement et nommé à l'aide d'un attribut `id`. Tout document qui est le contenu d'un cadre dans le jeu d'encadrement peut alors se référer à cet identifiant.
- Il est préférable de placer un élément `NOFRAMES` dans l'élément `FRAMESET` afin d'offrir un contenu de remplacement aux agents utilisateurs qui ne reconnaissent pas les cadres ou ne sont pas configurés pour afficher ces cadres.
- Vous pouvez insérer un cadre en ligne au sein d'un bloc de texte à l'aide de l'élément `IFRAME`. Un tel cadre ne peut jamais être redimensionné.
- D'une certaine façon, les problèmes de conception d'un jeu d'encadrement sont très proches de ceux rencontrés lors de la conception de tableaux. Un des principaux problèmes posés, comme avec les tableaux, concerne la taille de l'écran. Il est de nos jours raisonnable de considérer que l'écran de l'utilisateur possède une dimension au moins égale à 800 pixels de large pour 600 pixels de haut.

Chapitre 8

Feuilles de style

Introduction 279
Attribut style 281
Élément STYLE 282
Feuille de style externe 294
Propriétés de feuille de style 306
Feuille de style en cascade 323
Feuille de style auditive, pour un public particulier 328
Résumé 335

Vous avez remarqué dans les chapitres précédents que les attributs et certains éléments relatifs à la mise en forme et à la présentation de pages Web étaient fréquemment, signalés comme **déconseillés** en faveur du recours aux feuilles de style.

Lors de l'apparition du Web, les principaux utilisateurs étaient essentiellement des scientifiques, des enseignants et des militaires, plus concernés par le contenu des documents que par leur présentation. Le développement fulgurant du World Wide Web et l'arrivée de personnes issues d'horizons très différents a mis rapidement en évidence que les limites de HTML constituaient une source de frustration perpétuelle. Nombreux furent alors ceux qui durent contourner les limites de mise en forme d'HTML par des méthodes aussi diverses qu'astucieuses. Aussi louable qu'ait pu être le but d'améliorer la présentation des pages Web, les techniques employées pour ce faire possédaient des effets secondaires malheureux : leur fonctionnement n'était pas garanti et restait largement dépendant de divers facteurs, dont l'agent utilisateur. Les principales méthodes employées étaient :

- L'emploi d'extensions HTML propriétaires.
- La conversion du texte en image.
- Le recours à des images pour contrôler l'espacement.
- L'emploi de tableaux pour la mise en page.
- L'écriture d'un programme plutôt que le recours à HTML.

Ces techniques accroissent considérablement la complexité des pages Web, offrent peu de souplesse, souffrent de problèmes d'interopérabilité et constituent une épreuve pénible pour les personnes présentant des handicaps.

Les feuilles de style HTML représentent un progrès majeur qui développe les possibilités d'amélioration de l'aspect des pages Web. Elles résolvent tous ces problèmes en même temps qu'elles remplacent l'éventail limité des mécanismes de présentation HTML. Avec les feuilles de style, il devient facile de spécifier l'espacement entre les lignes de texte, l'indentation des lignes de texte, la couleur utilisée pour le texte et l'arrière-plan, la taille et le style de la police et quantité d'autres détails.

Les feuilles de style HTML peuvent être mises en œuvre de trois façons différentes :

- Au sein d'un élément.

- Dans l'élément `HEAD` du document.
- À l'aide d'une feuille de style externe.

Ces trois méthodes ne sont pas exclusives les unes des autres. Ce chapitre étudie les différentes modalités de mise en œuvre des feuilles de style.

Éléments et attributs étudiés dans ce chapitre :

```
STYLE
LINK
color, background-color, font-, text-, margin-, border-,
padding-, height, width, overflow, float, position, left,
top, z-index, clip, visibility, media
speak, spell-out, cue-, pause-, azimuth, elevation,
voice-family, volume, silent, speech-rate, richness,
stress
@media
```

8.1. Introduction

Une des motivations sous-jacentes aux feuilles de style, outre celles citées plus haut, était d'améliorer la séparation de la logique de structure (définie par les éléments HTML) de celle de la présentation (désormais définie par les feuilles de style). Cette démarche de séparation des différentes couches de logique, de plus en plus fréquente en matière de programmation, s'inspire du modèle MVC (voir encadré).



Séparation de la logique et de la présentation : le modèle MVC (Model-View-Controller)

Le modèle MVC est un modèle de conception logicielle largement répandu, initialement créé dans les années 1980 par Xerox PARC pour Smalltalk-80. Il a été plus récemment recommandé comme modèle pour la plate-forme J2EE de Sun et gagne fortement en popularité auprès des développeurs ColdFusion et PHP. C'est un outil fort utile au développeur, quel que soit le langage utilisé, même s'il présente quelques inconvénients.

Le modèle de conception MVC impose la séparation entre les données, les traitements et la présentation. Toute application est ainsi divisée en trois composants fondamentaux : le modèle, la vue et le contrôleur. Chacun de ces composants possède un rôle bien défini.

La **vue** est l'interface avec laquelle interagit l'utilisateur. Pour les applications Web, c'était historiquement une interface HTML, mais cela peut également être une interface Macromedia Flash ou d'autres langages de balises comme XHTML, XML/XSL, WML et les services Web. Le



MVC sait gérer l'utilisation de différentes vues pour une même application. Aucun traitement n'est effectué dans la vue : celle-ci ne sert qu'à afficher les données et à permettre à l'utilisateur d'agir sur celles-ci. Le deuxième composant du MVC, le **modèle**, représente les données et les règles de travail (là où s'effectuent les traitements). Les bases de données en font partie, de même que des objets comme les EJB et composants ColdFusion. Les données renvoyées par le modèle sont indépendantes de la présentation : le modèle ne réalise aucune mise en forme. Un même modèle peut afficher ses données dans plusieurs vues. Enfin, le **contrôleur** interprète les requêtes de l'utilisateur et appelle le modèle et la vue nécessaires pour répondre à celles-ci. Lorsque l'utilisateur clique sur un lien ou soumet un formulaire HTML, le contrôleur ne produit rien et n'effectue aucun traitement : il intercepte simplement la requête pour déterminer les modèles et vues qui doivent être associés.

Pour résumer, une requête utilisateur est interprétée par le contrôleur, qui détermine les portions du modèle et de la vue qui doivent être appelées. Le modèle gère les interactions avec les données et applique les règles de travail, puis renvoie les données. Le contrôleur sélectionne enfin une vue et lui transmet les données.

En appliquant ce modèle à HTML, la feuille de style serait la vue, l'agent utilisateur le contrôleur et le document HTML le modèle.

La spécification HTML 4.01 ne lie pas HTML à un langage de style particulier. Cela permet l'utilisation d'un panel de langages (les plus simples pour la majorité des utilisateurs et les plus complexes pour une minorité d'utilisateurs aux besoins très spécialisés). Le langage le plus fréquemment employé est CSS (*Cascading Style Sheet*), fondé sur la Recommandation W3C CSS niveau 2 (dont la version révisée au 7 juin 2005 est disponible en français à l'adresse **www.yoyodesign.org/doc/w3c/css2/cover.html**) et la version 2.1 *release candidate* (disponible en anglais à l'adresse **http://www.w3.org/TR/CSS21/**). C'est celui dont nous nous servirons dans cet ouvrage. La spécification CSS niveau 3 est en cours d'élaboration (voir **http://www.w3.org/Style/CSS/current-work**). Souvenez-vous toutefois que d'autres langages de feuille de style sont possibles en HTML.

Pour employer dans vos pages des feuilles de style, vous devez spécifier le langage employé, à l'aide d'un élément `META`.

1 Ouvrez le fichier *index.html* dans le Bloc-Notes.

2 Modifiez le numéro de version en 3.8.1 (il s'agit là encore d'une modification majeure) :

```
<META name="version" content="3.8.1">
```

3 Ajoutez juste en dessous, avant la balise `</HEAD>`, l'élément `META` suivant :

```
<META http-equiv="Content-Style-Type"
      content="text/css">
```

4 Enregistrez le fichier *index.html* sans modifier son nom.

5 Répétez les étapes 1 à 4 pour les fichiers *passions.html*, *region.html*, *famille.html* et *barrenav.html*.

6 Ouvrez le fichier *accueil.html*. Modifiez son numéro de version, ajoutez le nouvel élément `META`, puis gardez ce fichier ouvert dans le Bloc-Notes.

Comme vous devez l'avoir compris, il s'agit d'une ligne essentielle à ajouter désormais systématiquement dans l'élément `HEAD` de tout fichier HTML.

8.2. Attribut style

La façon la plus simple de définir un style dans une page, pour rester en conformité avec la spécification HTML 4.01, consiste à le faire en ligne à l'aide de l'attribut `style`. Cet attribut peut être employé avec tous les éléments HTML sauf `BASE`, `BASEFONT`, `HEAD`, `HTML`, `META`, `PARAM`, `SCRIPT`, `STYLE` et `TITLE`.

Passons à la pratique.

1 Revenez au fichier *index.html*, qui devrait être ouvert dans votre Bloc-Notes.

2 Modifiez comme suit la ligne de titre (l'élément `H1`), en ajoutant un attribut `style` :

```
<H1 style="font-size: 32pt; text-align: center; color:
✂ green">Ma page d'accueil</H1>
```

3 Enregistrez le fichier sans modifier son nom.

Lancez votre navigateur, puis ouvrez le fichier *index.html* (voir Figure 8.1).

La syntaxe de la valeur de l'attribut `style` est déterminée par le langage de feuille de style par défaut. Par exemple, pour un style en ligne CSS, vous devez employer la syntaxe de bloc de déclaration CSS. La syntaxe générale CSS de l'attribut `style` est donc :



Figure 8.1 : *Page d'accueil, avec titre principal, modifiée à l'aide d'un attribut style.*

```
style ="propriété1 : valeur; propriété2 = valeur; …
        propriétén = valeur"
```

Pour spécifier les informations de style de plusieurs éléments, mieux vaut préférer l'élément `STYLE`.

8.3. Élément STYLE

L'élément `STYLE` permet de rassembler des règles de style dans l'en-tête du document, qui vont ainsi s'appliquer à la totalité des éléments concernés du document. HTML autorise un nombre quelconque d'éléments `STYLE` dans la section `HEAD` d'un document.

La syntaxe des données de style dépend du langage de feuille de style.

Certaines implémentations de feuille de style peuvent autoriser une plus grande variété de règles dans l'élément `STYLE` que dans l'attribut `style`. D'une façon générale, CSS distingue les **sélecteurs** (la définition d'un élément) et les **définitions** (les propriétés de présentation affectées à cet élément). CSS permet de déclarer les règles à l'intérieur d'un élément `STYLE` pour :

- Toutes les instances d'un élément HTML particulier (par exemple, tous les éléments `P`, tous les éléments `H1`, etc.).

  ```
  H1 {définitions}
  ```

- Toutes les instances d'un élément HTML appartenant à une classe particulière (par exemple, les éléments dont l'attribut `class` possède une valeur spécifiée).

```
*[class=valeur] {définitions}
```

- Les instances uniques d'un élément HTML (par exemple, un élément dont l'attribut `id` possède une valeur spécifiée).

```
élément[id=valeur] {définitions}
```

Il ne s'agit là toutefois que d'exemples simplissimes : la version 3 des sélecteurs CSS (**http://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html**) propose des sélecteurs nombreux et très sophistiqués, capables de répondre désormais à pratiquement toutes les situations (reportez-vous au tableau 13.12 du Chapitre 13, « Annexes »). Les possibilités d'enrichissement d'un contenu à l'aide d'une feuille de style sont désormais extrêmement riches. En outre, CSS permet de combiner à l'aide d'opérateurs les sélecteurs simples, ce qui élargit encore le champ des possibilités.



Sensibilité à a casse dans les sélecteurs

Dans les sélecteurs, la sensibilité à la casse des noms d'éléments et d'attributs ainsi que des valeurs des attributs dépend du langage du document. Par exemple, les noms d'éléments sont sensibles à la casse en XML mais pas en HTML.

Les règles de préséance et d'héritage des règles de style dépendent du langage de feuille de style.

Examinons successivement ces trois possibilités.

Définition du style de la totalité des occurrences d'un élément donné

L'élément `STYLE` permet de définir facilement l'aspect d'éléments HTML spécifiés. Voici comment procéder :

1. Ouvrez dans le Bloc-Notes le fichier *accueil.html*.
2. Enregistrez-le en l'état sous le nom *accueil_3_8_1.html*. Modifiez ensuite son numéro de version :

```
<META name="version" content="3.8.2">
```

3 Avant la balise de fermeture de l'élément `HEAD`, ajoutez un élément `STYLE` doté du contenu suivant :

```
<STYLE type="text/css">
   H1 {border-width: 1; border: solid;
       text-align: center;
       font-size: 32pt; color: green}
   H2 {font-size: 20pt; text-align: center}
   P {font-size: 16pt; text-align: center}
   UL {font-size: 16 pt; text-align: center}
   LI {font-size: 14pt; text-align: left;
       margin-left: 40%}
   ADDRESS {text-align: center}
</STYLE>
</HEAD>
```

4 Supprimez l'attribut `style` de l'élément `H1` ainsi que l'attribut `align` de l'élément `H2` (puisque vous avez défini ces styles dans l'en-tête), ainsi que les balises de l'élément `DIV` (les styles des éléments contenus sont également définis dans l'en-tête). Comme vous avez défini un cadre autour de l'élément `H1`, vous pouvez également supprimer la règle horizontale qui le suit (l'élément `HR`).

5 Enregistrez le fichier d'abord sous le nom *accueil_3_8_2.html*, puis sous le nom *accueil.html*.

Lancez votre navigateur, puis ouvrez le fichier *index.html*. Remarquez que vous lancez *index.html* puisque celui-ci charge automatiquement *accueil.html* dans un jeu d'encadrement (voir Figure 8.2).

Cette page ne diffère que très peu de la précédente. Nous examinerons plus en détail dans la suite de ce chapitre les différents attributs qu'il est possible d'employer dans une feuille de style, mais regardez déjà ce que nous avons fait ici :

- Dans l'élément `H1`, la propriété `border-width` crée un cadre épais d'un pixel, la propriété `border` définissant son aspect (ligne pleine).
- La propriété `color`, déjà rencontrée, définit la couleur du texte.
- La propriété `font-size`, également déjà rencontrée, définit la taille de la police. Nous avons employé des tailles décroissantes selon les cas.

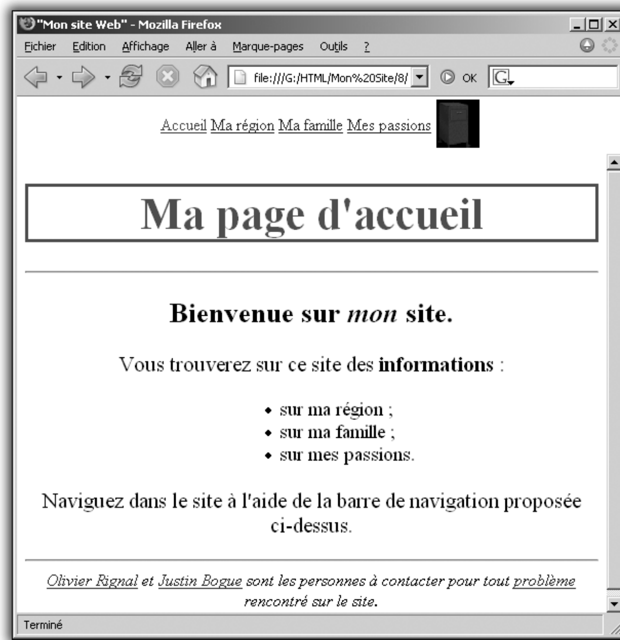


Figure 8.2 : *Page d'accueil avec, cette fois, recours à un élément STYLE.*

- La propriété `text-align`, déjà rencontrée aussi, définit l'alignement du texte. Les éléments `H1`, `H2`, `P`, `UL` et `ADDRESS` sont définis comme devant être centrés à l'écran. En revanche, l'élément `LI` doit être aligné à gauche.
- Enfin, la propriété `margin-left: 40%` de l'élément `LI` définit une marge gauche égale à 40 % de l'espace disponible : l'effet obtenu avec cette marge et l'alignement à gauche est plus satisfaisant que le centrage précédent.

Grâce à la définition des styles dans l'élément `HEAD`, toute occurrence des éléments définis sera rendue de façon homogène, selon les critères spécifiés. Un énorme gain de temps et la garantie de disposer d'un effet homogène !

Effet homogène, mais peut-être trop. Vous pourriez souhaiter définir des types de présentation particuliers pour certaines occurrences d'un élément précis . Cela est possible, grâce à l'emploi de l'attribut `classe`.

Définition du style de certaines occurrences d'un élément donné

Souvenez-vous : nous avons prévu dans la page d'accueil une place pour des remarques légales, mais sans encore les créer. Si nous les mettons dans un élément P, elles possèderont une taille de police et un centrage automatiques, compte tenu de la feuille de style actuelle. Il faut donc indiquer que, pour les éléments P qui appartiennent à une classe particulière, l'aspect doit être autre. Procédez comme suit :

1. Revenez dans le Bloc-Notes au fichier *accueil.html*.
2. Modifiez son numéro de version :

```
<META name="version" content="3.8.3">
```

3. Ajoutez dans l'élément STYLE, juste après la définition de l'élément P, une nouvelle définition :

```
   P {font-size: 16pt; text-align: center}
   P.legal {font-size: 8pt; font-variant:
     small-caps}
   UL {font-size: 16 pt; text-align: center}
```

4. Remplacez le commentaire <!–– Des remarques légales ne seraient pas superflues ––> par l'élément P suivant :

```
<P class="legal">&copy; 2005 votrenom -
Tous droits r&eacute;serv&eacute;s</P>
```

5. Enregistrez le fichier d'abord sous le nom *accueil_3_8_3.html*, puis sous le nom *accueil.html*.

```
<HEAD>
 <STYLE type="text/css">
   H1.maclasse {border-width: 1; border: solid;
                text-align: center}
 </STYLE>
</HEAD>
<BODY>
 <H1 class="maclasse"> Cet H1 est touch&eacute;
  par notre style </H1>
 <H1> Celui-ci ne l'est pas </H1>
</BODY>
```

Ouvrez le fichier index.html dans votre navigateur (voir Figure 8.3).

Le paragraphe des remarques légales, bien qu'étant un élément P, est présenté différemment puisque nous avons spécifié que cet élément appartenait à la classe legal.



Figure 8.3 : *Définition d'un style pour une classe d'éléments particulière*

Remarquez l'attribut `type` de l'élément `STYLE`. Sa syntaxe est :

```
type = "type_de_contenu"
```

Cet attribut spécifie le langage de feuille de style par défaut du contenu de l'élément. Il surclasse, lorsqu'il est présent et qu'elle est spécifiée, la feuille de style par défaut (définie par `<META http-equiv= "Content-Style-Type" content="type_de_contenu">` dans `l'élément HEAD`). Le langage de feuille de style est spécifié comme un type de contenu (par exemple, `"text/css"`). Si la définition de la feuille de style par défaut est absente, il est indispensable de fournir une valeur pour cet attribut : il est dépourvu de valeur par défaut.

Définition du style d'une unique occurrence d'un élément donné

Vous pourriez également souhaiter qu'un élément unique dans votre, voire vos pages respecte un style particulier. Vous avez alors recours à l'identifiant de cet élément (l'attribut `id`), lequel doit être unique dans le document. La syntaxe est la suivante :

```
<STYLE>
#monid {propriété1 :valeur; propriété2: valeur;...
        propriétén: valeur}
</STYLE>
```

La mise en œuvre serait comme suit :

```
<P id="monid"> Ce paragraphe est soumis au style </P>
<P> mais pas celui-la. </P>
```

Vous pourriez vous poser la question de l'intérêt de définir dans la feuille de style d'en-tête un élément unique : ne serait-il pas plus simple de définir un attribut `style` dans cet élément ?

La réponse tient à ce que nous avons déjà souligné à plusieurs reprises : HTML doit être un langage de spécification de contenu. Son but est avant tout la structure et non la présentation. Mieux vaut donc séparer au maximum la logique structurelle (les éléments HTML et leur contenu) de la logique de présentation (la feuille de style).

Éléments DIV et SPAN

Vous avez compris qu'il était possible de définir des informations de style pour pratiquement tous les éléments HTML. Deux éléments, `DIV` et `SPAN`, méritent cependant des mentions particulières.

La plupart des éléments HTML possèdent déjà une mise en forme « intégrée ». Un paragraphe de texte place une ligne avant et après lui-même, et utilisera une police différente de celle des autres éléments dans certains navigateurs.

Dans la mesure où les feuilles de style sont supposées combiner les indications structurelles et les indications de mise en forme, il fallait créer des éléments « neutres ». L'élément `DIV` est un élément de type bloc, ce qui signifie qu'une ligne sera insérée avant et après lui. En termes de structure, il ne correspond qu'au souhait de lui voir présenter un aspect particulier, différent du comportement des éléments de structure classique. Il ne possède pourtant aucune autre mise en forme particulière par défaut, si bien que le texte qui y figure apparaîtra comme texte par défaut (sauf s'il est spécifié autrement).

`SPAN` fonctionne de la même façon, si ce n'est qu'il s'agit d'un élément ligne. Ces deux éléments peuvent être définis dans une feuille de style comme n'importe quel autre élément.

En conséquence, vous pouvez utiliser ces éléments pour insérer des styles conçus entièrement sur la structure de la page, plutôt que selon ses éléments de conception. Par exemple, si la première section de votre

document contient les informations les plus importantes, placez-les dans un élément DIV auquel vous attribuez un style particulier : une couleur spéciale, une police caractéristique ou un texte en italique. Tout son contenu, y compris les titres et les images, hériteront du style DIV.

De ce fait, l'emploi de ces éléments conjointement aux feuilles de style permet un développement HTML pratiquement illimité, particulièrement quand ils sont employés avec les attributs class et id : cela permet de développer potentiellement une infinité de types de présentation différents. Par exemple :

```
<STYLE type="text/css">
   H1 {border-width: 1; border: solid; text-align: center;
       font-size: 32pt; color: green}
   P {font-size: 16pt; text-align: center}
   DIV {font-size: 16pt; text-align: center}
   DIV.it {font-style: italic}
   DIV.g {font-weight: bold}
   DIV.itg {font-style: italic; font-weight: bold}
   SPAN.it {font-style: italic}
   SPAN.g {font-weight: bold}
   SPAN.itg {font-style: italic; font-weight: bold}
</STYLE>
```

Regardez le fichier suivant (volontairement minimaliste), dans lequel les parties importantes sont présentées en gras :

Listing 8-1 : Fichier STYLEDIVSPAN.html

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>"El&eacute;ment STYLE avec des
    &eacute;l&eacute;ments DIV et SPAN "</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
      charset=iso-8859-1">
    <META http-equiv="Content-Style-Type" content="text/css">
    <STYLE type="text/css">
      H1 {text-align: center}
      P {font-size: 16pt; text-align: center}
      DIV {font-size: 16pt; text-align: center}
      DIV.it {font-style: italic}
      DIV.g {font-weight: bold}
      DIV.itg {font-style: italic; font-weight: bold}
      SPAN.it {font-style: italic}
      SPAN.g {font-weight: bold}
      SPAN.itg {font-style: italic; font-weight: bold}
    </STYLE>
```

```
  </HEAD>
  <BODY>
    <H1>El&eacute;ment <CODE>STYLE</CODE> avec
       des &eacute;l&eacute;ments <CODE>DIV</CODE>
       et <CODE>SPAN</CODE></H1>
    <P>Ceci est un paragraphe normal.</P>
    <DIV class="it">Ceci est un paragraphe en italiques</DIV>
    <DIV class="g">Ceci est un paragraphe en gras</DIV>
    <DIV class="itg">Ceci est un paragraphe en italiques gras
       </DIV>
    <P>Ceci est un paragraphe normal, avec quelques mots
    <SPAN class="it">en italiques</SPAN>,
 quelques-uns <SPAN class="g">en gras</SPAN> et d'autres
 <SPAN class="itg">en italiques gras</SPAN>.</P>
  </BODY>
</HTML>
```

Examiné dans votre navigateur, ce fichier affiche ce qui est présenté dans la figure suivante.

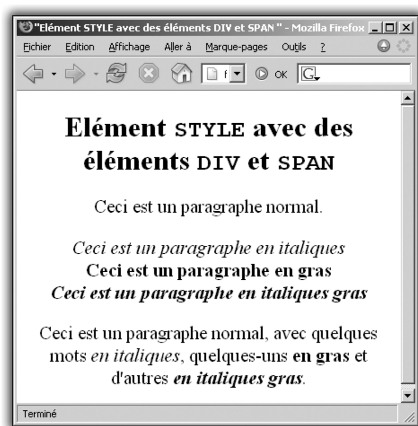


Figure 8.4 :
Définition d'un style pour des classes d'éléments DIV et SPAN

Le gain de saisie semble ici relativement faible, mais imaginez que vous ayez à modifier la taille et le type de la police, sa couleur, l'alignement, etc.

Remarquez en outre dans cette figure que les éléments `DIV` des classes `it`, `g` et `itg` héritent du style défini pour l'élément générique `DIV` : ici, la taille de police et le centrage. Nous reviendrons par la suite sur cette notion d'héritage, mais sachez dès à présent qu'un style hérite des caractéristiques par défaut de son style parent.

Attribut media

HTML permet aux auteurs de spécifier les documents qui exploitent les caractéristiques du médium sur lequel le document doit être restitué (par exemple : écran graphique, écran de télévision, appareil de poche, navigateur à synthèse vocale, appareil tactile braille, etc.). Cela ouvre aux utilisateurs l'accès aux pages Web au moyen d'une grande variété d'appareils.

Les feuilles de style s'appliquent à un médium spécifique ou à un groupe de médias. Une feuille de style prévue pour un écran peut être éventuellement utilisable pour l'impression, mais sera de peu d'utilité pour un navigateur à synthèse vocale. Cette spécification permet de définir les catégories générales de médias auxquelles peut s'appliquer une feuille de style donnée. Ceci permet aux agents utilisateurs d'éviter la recherche de feuilles de style inappropriées. Les langages de feuilles de style peuvent inclure des mécanismes décrivant les dépendances aux médias dans une même feuille de style.

L'attribut `media` de l'élément `STYLE` spécifie le média de destination désiré pour les informations de style. Cela peut être un descripteur de média unique ou une liste de descripteurs, séparés par des virgules. La valeur par défaut de cet attribut est `screen`, soit un écran d'ordinateur (le cas le plus fréquent). Sa syntaxe est la suivante :

```
media = "descripteur_médium1, descripteur_médium2, ...
         descripteur_médiumn"
```

Les valeurs possibles et les significations de celles-ci sont présentées dans le tableau suivant.

Tableau 8.1 : *Descripteurs de médias*

Descripteur de médium	But
`screen`	Écrans d'ordinateurs non paginés
`tty`	Périphérique utilisant une grille de caractères fixe, comme un télex.
`tv`	Périphériques de type télévision
`projection`	Vidéoprojecteurs
`handheld`	Périphériques tenus à la main (en principe écran de petite taille et faible bande passante)

Tableau 8.1 : Descripteurs de médias	
Descripteur de médium	**But**
`print`	Matériaux opaques paginés et documents vus à l'écran en mode Aperçu avant impression
`braille`	Périphérique à rendu tactile braille
`embossed`	Imprimantes à pagination braille
`speech`	Synthétiseurs de parole. Anciennement `aural` en CSS 2.0.
`all`	Tous périphériques

Par exemple, les déclarations des exemples suivants s'appliquent aux éléments `H1`. Pour une projection dans une réunion de travail, toutes leurs instances apparaîtront en bleu. Pour une impression papier, toutes les instances seront centrées.

```
<STYLE type="text/css" media="projection">
   H1 {color: blue}
</STYLE>
<STYLE type="text/css" media="print">
   H1 {text-align: center}
</STYLE>
```

Le choix du support est particulièrement intéressant quand il s'applique aux feuilles de style externes, dans la mesure où les agents utilisateurs peuvent économiser du temps en ne chargeant à partir du réseau que les feuilles de style qui concernent le périphérique courant. Nous reviendrons donc sur ce sujet lors de l'étude des feuilles de style externes.

Masquage du contenu de l'élément STYLE

Les agents utilisateurs qui ne reconnaissent pas les feuilles de style ou le langage de feuille de style spécifique employé par un élément `STYLE` doivent dissimuler le contenu de l'élément `STYLE`. La restitution du contenu de cet élément en tant que partie du texte du document constitue une erreur. Certains langages de feuilles de style gèrent une syntaxe pour dissimuler le contenu aux agents utilisateurs non conformes. C'est le cas de CSS.

1 Revenez dans le Bloc-Notes au fichier *accueil.html*.

2 Modifiez son numéro de version :

```
<META name="version" content="3.8.4">
```

3 Modifiez comme suit l'élément `STYLE` :

```
<STYLE type="text/css">
   <!--
   H1 {border-width: 1; border: solid; text-align: center;
       font-size: 32pt; color: green}
   H2 {font-size: 20pt; text-align: center}
   P {font-size: 16pt; text-align: center}
   P.legal {font-size: 8pt; font-variant: small-caps}
   UL {font-size: 16 pt; text-align: center}
   LI {font-size: 14pt; text-align: left; margin-left: 40%}
   ADDRESS {text-align: center}
   -->
</STYLE>
```

5 Enregistrez le fichier d'abord sous le nom *accueil_3_8_4.html*, puis sous le nom *accueil.html*.

Ouvrez le fichier *index.html* dans votre navigateur. Vous devriez obtenir quelque chose de similaire à ce qui est présenté dans la figure suivante,



Figure 8.5 : *Le masquage optionnel du contenu de l'élément STYLE reste sans conséquence si l'agent utilisateur reconnaît cet élément*

soit quelque chose de parfaitement identique à ce qui était présenté dans celle-ci.

La seule différence est qu'un agent utilisateur qui ne reconnaît pas l'élément `STYLE` n'affichera pas son contenu comme du texte.



Figure 8.6 : *Définition d'un style pour une classe d'éléments particulière*

En poussant le raisonnement à propos de la séparation de la structure et de la présentation à sa conclusion logique, la séparation doit être encore plus drastique et les styles définis dans une ou plusieurs feuilles de style externes. C'est ce que nous allons examiner maintenant.

8.4. Feuille de style externe

Outre l'objectif de séparation de la structure et de la présentation, d'excellentes raisons incitent à employer des feuilles de style :

- Il est possible de partager des feuilles de style entre plusieurs documents et/ou sites.
- Une feuille de style peut être modifiée sans qu'il soit nécessaire de modifier le document.
- Les agents utilisateurs peuvent charger les feuilles de style sélectivement, en fonction des descriptions des médias.

Cela a toujours été et reste une des recommandations fondamentales de la spécification CSS. Le principal avantage est qu'il suffit de modifier une feuille de style externe pour modifier l'aspect général du site. Cela est fort pratique si vous utilisez un modèle général, chaque page pouvant toujours, si nécessaire, posséder un aspect particulier grâce aux classes et aux identifiants.

Comme cela a été mentionné plus tôt, il est possible de lier une feuille de style externe à n'importe quel document Web. Une feuille de style

externe s’adapte particulièrement bien aux sites de grande taille : il suffit de modifier un fichier pour affecter l’intégralité du site. En combinaison avec un modèle de page Web, cela facilite l’organisation et la standardisation du site.

Une feuille de style n’est rien d’autre qu’un fichier texte qui indique à l’agent utilisateur comment restituer votre code HTML. Elle complète les éléments standard : si l’élément `H1` signale un titre, la balise `<H1 style="color: green;">` correspond à un titre vert. Une feuille de style externe emmène HTML un pas plus avant dans la séparation réussie entre logique de travail et logique de présentation.

Pour créer une feuille de style externe, débutez par un fichier texte ordinaire. Il n’est pas besoin d’utiliser un quelconque élément ou balise : saisissez simplement vos propriétés à la volée, exactement comme si elle étaient incluses dans l’élément `STYLE` d’un fichier HTML.

Comme nous l’avons vu précédemment, un style se compose de propriétés qui possèdent des valeurs. Pour le paragraphe vert, la propriété est `color` et sa valeur est `green`. Vous allez créer votre première feuille de style.

1. Ouvrez un document vierge dans le Bloc-Notes. Saisissez ce qui suit (vous pouvez également effectuer un copier-coller du contenu de l’élément `STYLE` du fichier *accueil.html*, puisque les informations sont identiques) :

   ```
   /* style.css : feuille de style pour mon site */
   H1 {border-width: 1; border-style: solid; text-align: center;
      font-size: 32pt; color: green}
   H2 {font-size: 20pt; text-align: center}
   P {font-size: 16pt; text-align: center}
   P.legal {font-size: 8pt; font-variant: small-caps}
   UL {font-size: 16 pt; text-align: center}
   LI {font-size: 14pt; text-align: left; margin-left: 40%}
   ADDRESS {text-align: center}
   ```

   Comme vous le voyez, un document `.css` ne possède ni ouverture ni fermeture. Les commentaires sont encadrés de symboles `/*` et `*/` : un vieux souvenir de syntaxe de programmation.

2. Enregistrez ce fichier sous le nom *style.css*.

   Attention, un fichier de feuille de style doit toujours comporter une extension *.css*.

3 Ouvrez le fichier *accueil.html*.

4 Modifiez son numéro de version :

```
<META name="version" content="3.8.5">
```

5 Remplacez l'élément `STYLE` par l'élément `LINK` suivant :

```
<LINK rel="STYLESHEET" type="text/css"
      href="style.css">
```

6 Enregistrez le fichier d'abord sous le nom *accueil_3_8_5.html*, puis sous le nom *accueil.html*.

Ouvrez le fichier `index.html` dans votre navigateur. Vous devriez obtenir quelque chose de similaire à ce qui est présenté dans la figure suivante, soit encore une fois quelque chose de parfaitement identique à ce qui était présenté Figure 8.3 ou Figure 8.5. La seule différence est que les styles sont désormais définis dans une feuille de style externe.



Figure 8.7 : *Le site employant désormais une feuille de style externe*

Pour tirer un profit maximum de notre feuille de style, il faut l'appliquer aux autres pages de notre site, en complétant si besoin les différentes spécifications. Commençons par le fichier le plus simple, *passions.html*.

1 Ouvrez *passions.html* dans le Bloc-Notes.

2 Modifiez son numéro de version :

```
<META name="version" content="3.8.1">
```

3 Ajoutez l'élément `LINK` identique à celui du fichier *accueil.html*.

```
<LINK rel="STYLESHEET" type="text/css"
      href="style.css">
```

4 Supprimez les attributs `align` des éléments `H1` et `H2`, puisqu'ils sont désormais superflus.

5 Enregistrez le fichier sans modifier son nom.

Ouvrez le fichier *index.html* dans votre navigateur, puis cliquez dans la barre de navigation sur **Mes passions**. Vous devriez obtenir quelque chose de similaire à ce qui est présenté dans la figure suivante, c'est-à-dire un aspect beaucoup plus proche de celui de la page d'accueil.



Figure 8.8 :
Page Mes passions, employant désormais la feuille de style externe.

Les pages *Ma région* et *Ma famille* sont un peu plus complexes, puisqu'elles font appel à des tableaux. Nous devons donc définir un style de tableau dans la feuille de style.

1 Ouvrez *style.css* dans le Bloc-Notes.

2 Ajoutez les définitions de style suivantes à la fin du fichier :

```
TABLE {width: 100%; border-width: 2;
       border-style:double; text-align: center}
CAPTION {font-size: 24px; font-weight: bold;
       text-align: center;}
TH {font-size: 20pt; border-width: 1;
       border-style: solid; text-align: center}
TD {font-size: 14pt; border-width: 1;
       border-style: solid; text-align: center}
TD.droit {font-size: 14pt; text-align: right}
```

3 Enregistrez le fichier sans modifier son nom.

4 Ouvrez *region.html* dans le Bloc-Notes.

5 Modifiez son numéro de version :

```
<META name="version" content="3.8.1">
```

6 Ajoutez l'élément `LINK` identique à celui des fichiers *accueil.html* et *passions.html* :

```
<LINK rel="STYLESHEET" type="text/css"
      href="style.css">
```

7 Supprimez les attributs `align` et `border` du premier tableau, et les attributs `align`, `border` et `width` du second tableau.

8 Remplacez les attributs `align="right"` de ses trois éléments `TD` par un attribut `class="droit"`, comme dans l'exemple suivant :

```
<TH>Temp. Max.</TH><TD class="droit">18 °</TD>
```

9 Enregistrez votre fichier sans modifier son nom.

Ouvrez le fichier *index.html* dans votre navigateur, puis cliquez dans la barre de navigation sur **Ma région**.

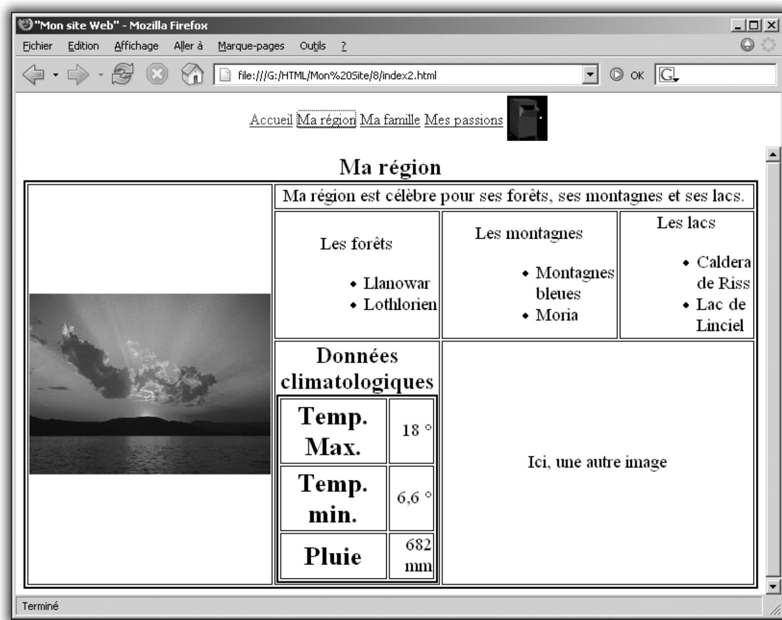


Figure 8.9 : *Page Ma région, employant désormais la feuille de style externe.*

Similaire, mais pas identique : nous avons en réalité légèrement modifié le contenu de ce fichier, en y ajoutant du contenu et en remplaçant les caractères accentués par les entités adéquates. Voici son listing complet, les modifications étant signalées en gras.

Listing 8-2 : region.html version 3.8.1

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE>Ma r&eacute;gion</TITLE>
    <META name="author" content="Fabrice Lemainque">
    <META name ="version" content="3.8.1">
    <META http-equiv="Content-Style-Type" content="text/css">
    <LINK rel="STYLESHEET" type="text/css" href="style.css">
  </HEAD>
  <BODY>
     <TABLE summary="Pr&eacute;sentation de ma r&eacute;gion">
      <CAPTION>Ma r&eacute;gion</CAPTION>
      <TR>
<TD rowspan="3" width="34%">

    <!-- Essayer l'image JPEG -->
    <OBJECT data="Images/coucher de soleil.jpg"
        type="image/gif" width="100%"
        title="Un coucher de soleil sur le lac de Linciel">
    <!-- Sinon, le texte  -->
    Coucher de soleil sur le lac de Linciel
    </OBJECT>
  </OBJECT>
</OBJECT>
        </TD>
        <TD colspan="3">Ma r&eacute;gion est célèbre
        pour ses for&ecirc;ts, ses montagnes et ses lacs.</TD>
      </TR>
      <TR>
        <TD>Les for&ecirc;ts
         <UL>
          <LI>Llanowar
          <LI>Lothlorien
         </UL>
        </TD>
        <TD>Les montagnes
         <UL>
          <LI>Montagnes bleues
          <LI>Moria
         </UL>
        </TD>
        <TD>Les lacs
         <UL>
           <LI>Caldera de Riss
           <LI>Lac de Linciel
         </UL>
        </TD>
      </TR>
      <TR>
        <TD>
```

```
          <TABLE>
            <CAPTION>Donn&eacute;es climatologiques</CAPTION>
            <TR>
              <TH>Temp. Max.</TH><TD class="droit">18 °</TD>
            </TR>
            <TR>
              <TH>Temp. min.</TH><TD class="droit">6,6 °</TD>
            </TR>
            <TR>
              <TH>Pluie</TH></TH><TD class="droit">682 mm</TD>
            </TR>
          </TABLE>
        </TD>
        <TD colspan="2">Ici, une autre image </TD>
      </TR>
    </TABLE>
   </BODY>
</HTML>
```

Occupez-vous maintenant du dernier fichier, *famille.html*.

1 Ouvrez *famille.html* dans le Bloc-Notes.
2 Modifiez son numéro de version :

```
<META name="version" content="3.8.1">
```

3 Ajoutez l'élément `LINK` identique à celui des fichiers *accueil.html*, *passions.html* et *region.html* :

```
<LINK rel="STYLESHEET" type="text/css" href="style.css">
```

4 Supprimez les attributs `align` de l'élément `H1`. Supprimez les attributs `border`, `align`, `cellspacing`, `cellpadding` et `width` du tableau.
5 Supprimez les attributs `valign` des éléments `TD` et remplacez les attributs `align="right"` par un attribut `class="droit"`, comme dans l'exemple suivant :

```
<TH>Temp. Max.</TH><TD class="droit">18 °</TD>
```

6 Enregistrez votre fichier sans modifier son nom.

Ouvrez le fichier *index.html* dans votre navigateur, puis cliquez dans la barre de navigation sur **Ma famille** (voir Figure 8.10).

Toutes nos pages présentent désormais un aspect homogène. Imaginez maintenant que vous vouliez soudainement modifier la couleur du cadre placé autour de l'élément `H1`, et définir un retrait à droite et à gauche de celui-ci. Rien de plus simple !



Figure 8.10 : *Page Ma famille, employant désormais la feuille de style externe.*

1 Ouvrez le fichier *style.css* dans le Bloc-notes.

2 Modifiez comme suit la définition du style de l'élément `H1` :

```
H1 {margin-left: 15%; margin-right: 15%; border-width: 1;
    border: solid; text-align: center;
    font-size: 32pt; color: blue}
```

3 Enregistrez votre fichier.

Rafraîchissez votre navigateur ou ouvrez à nouveau *index.html*. Cliquez successivement sur les liens **Ma famille** et **Mes passions**. Tous les éléments `H1` ont été modifiés.



Figure 8.11 : *Une unique modification du fichier de feuille de style se répercute sur tous les éléments H1*



Figure 8.12 :
Une unique modification du fichier de feuille de style se répercute sur tous les éléments H1

Une seule modification du fichier de feuille de style se répercute sur l'ensemble du site.



Figure 8.13 :
Une unique modification du fichier de feuille de style se répercute sur tous les éléments H1

Cet exemple est d'une simplicité déconcertante. Pour voir à quel point une feuille de style bien conçue peut profondément modifier l'aspect d'un site, visitez www.csszengarden.com/ et cliquez sur les différents modèles proposés dans le cadre de droite : comme le montrent les deux copies d'écran qui suivent, les différences peuvent être phénoménales !



Figure 8.14 :
Exemple de l'effet des feuilles de style proposées par csszengarden.com

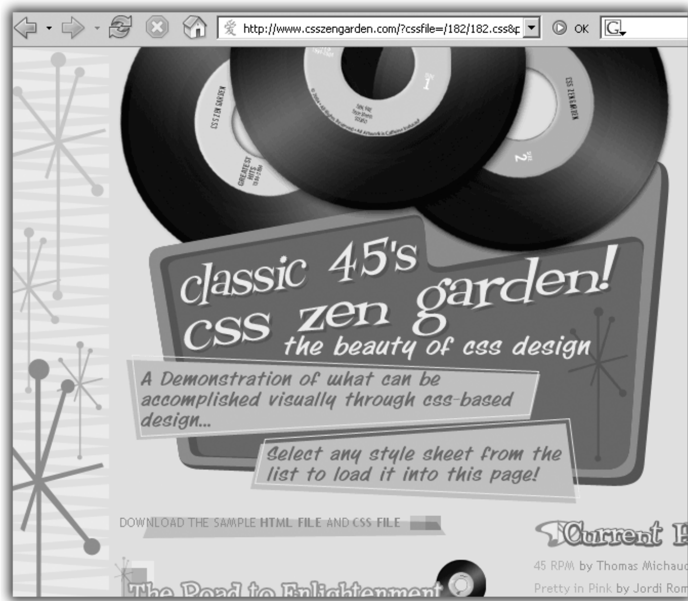


Figure 8.15 : *Autre exemple de l'effet des feuilles de style proposées par csszengarden.com*

Nous reviendrons bientôt plus en détail sur les différentes propriétés des feuilles de style. Pour le moment, penchons-nous plus avant sur l'élément `LINK`.

Élément LINK

Comme vous l'avez vu dans les exemples précédents, vous liez une feuille de style à un document HTML à l'aide de l'élément `LINK`. La syntaxe de celui-ci est la suivante :

```
<LINK rel="STYLESHEET" type="text/css"
      href="nomfeuilledestyle.css">
```

`href` est comme d'habitude une URL vers le fichier de feuille de style. Il semblerait toutefois que les URL absolues se comportent parfois de façon erratique : préférez les feuilles de style logées sur le serveur de votre site.

HTML permet d'associer un nombre quelconque de feuilles de style externes à un document. Le langage de feuille de style définit comment

plusieurs feuilles de style externes interagissent (par exemple, les règles de « cascade » de CSS).

Il est ainsi possible de spécifier plusieurs feuilles de style mutuellement exclusives, alors nommées « feuilles de style alternatives ». Cela permet d'offrir aux lecteurs plusieurs présentations d'un document. L'utilisateur peut sélectionner sa feuille de style favorite parmi celles-ci en fonction de ses préférences. Par exemple, une feuille de style pour la restitution des documents compacts dans une petite taille de police, ou une autre spécifiant une taille de police plus grande pour une meilleure lisibilité. Des feuilles alternatives peuvent viser des utilisateurs ou des médias spécifiques. Les agents utilisateurs doivent permettre aux utilisateurs de choisir entre plusieurs feuilles de style alternatives ainsi que de désactiver les feuilles de style.

Il est possible de spécifier une feuille de style préférée. Les agents utilisateurs doivent appliquer la feuille de style préférée de l'auteur, sauf si l'utilisateur sélectionne une autre alternative.

Plusieurs feuilles de style alternatives (y compris la préférée) peuvent être regroupées sous un seul nom de style. Lorsqu'un utilisateur sélectionne un style nommé, l'agent utilisateur doit appliquer toutes les feuilles de style qui portent ce nom, mais pas celles qui portent un nom différent. Vous verrez plus loin comment nommer un groupe de feuilles de style.

Il est également possible de spécifier des feuilles de style « persistantes » : elles sont appliquées par l'agent utilisateur en sus des éventuelles feuilles de style alternatives.

Les agents utilisateurs doivent respecter les descripteurs de médium lors de l'application d'une feuille de style. Ils doivent également permettre aux utilisateurs de désactiver entièrement les feuilles de style de l'auteur : aucune feuille de style n'est appliquée, qu'elle soit persistante ou alternative.

Les attributs de l'élément `LINK` jouent les rôles suivants :

- La valeur de l'attribut `rel` indique si la feuille de style est persistante, préférée ou alternative. Si cet attribut possède la valeur `"stylesheet"`, la feuille de style est persistante en l'absence d'attribut `title` et préférée si l'attribut `title` est spécifié. La feuille de style est en revanche alternative si l'attribut

rel possède la valeur "alternate stylesheet" et que la feuille est nommée à l'aide de l'attribut title.

- La valeur de l'attribut type indique le langage de la ressource reliée (la feuille de style). Cela évite à l'agent utilisateur de charger une feuille de style dans un langage de feuille de style non reconnu.
- La valeur de l'attribut href est un URI, qui indique la localisation du fichier de la feuille de style.

Les agents utilisateurs devraient fournir aux utilisateurs les moyens de passer en revue et de choisir les feuilles de style alternatives dans une liste. Mieux vaut employer la valeur de l'attribut title comme nom pour chaque option.

Vous pouvez également employer un élément META pour définir la feuille de style préférée du document. Par exemple, pour définir comme feuille de style préférée une feuille nommée "favorite", placez ce qui suit dans l'élément HEAD :

```
<META http-equiv="Default-Style" content="favorite">
```

Si plusieurs déclarations META spécifient la feuille de style préférée, seule la dernière est retenue. En revanche, si plusieurs éléments LINK spécifient une feuille de style préférée, seul le premier est retenu. Les feuilles de style préférées au moyen d'éléments META ont priorité sur celles spécifiées avec l'élément LINK.

Prise en compte du médium

Pour tenir compte du médium dans la liaison d'une feuille de style, servez -vous de l'attribut media dans l'élément LINK. Par exemple :

```
<LINK rel="stylesheet" type="text/css"
     media="print, handheld"
     href="styles.css">
```

Cet élément spécifie que cette feuille de style s'applique aux périphériques d'impression et aux périphériques tenus à la main (comme les téléphones portables et les assistants électroniques personnels).

Vous pouvez toutefois également utiliser la règle @media à l'intérieur d'une feuille de style afin que celle-ci puisse transmettre des informations à différents types de médias.

```
@media print {
              BODY {font-size: 10pt}
              }
@media screen {
               BODY {font-size: 12pt}
               }
```

Reportez-vous au tableau de la page 291 pour voir la liste des descripteurs de médium autorisés.

8.5. Propriétés de feuille de style

Après avoir vu comment créer une feuille de style, tant locale qu'externe, il est temps d'examiner plus en profondeur les propriétés qu'il est possible de définir dans une feuille de style. Souvenez-vous que, si HTML n'est en rien lié à un langage de feuille de style spécifique, nous traitons uniquement ici du langage CSS.

Propriétés de modification de texte et de police

Les propriétés les plus simples concernent la présentation du contenu textuel d'un document

Depuis son invention, l'élément `FONT` posait quelques problèmes aux concepteurs de pages Web. Il reste très mal supporté par les anciens navigateurs et entre parfois en conflit avec l'élément `BODY`. Les feuilles de style résolvent fort adroitement les problèmes posés par `FONT`. Les utilisateurs seront dans un proche avenir en mesure de définir une feuille de style par défaut pour leur navigateur, qui contiendra non seulement les couleurs et les tailles de police mais également les espacements, les bordures et les indentations. Ces éléments peuvent déjà tous être spécifiés à l'aide d'une feuille de style.

Les propriétés signalées ici peuvent être appliquées à tout élément qui possède un contenu texte.

Pour modifier la couleur d'une portion de texte, employez la propriété `color`. Par exemple :

```
<P style="color: #990000 ;">
```

place le paragraphe en rouge sombre (`#990000`).

Assurez-vous de bien fermer tout élément ayant recours à des styles, faute de quoi ceux-ci s'appliqueront à l'élément suivant.

Pour modifier la couleur de fond, utilisez `background-color`

```
<P style="color: #990000 ; background-color: #66FFFF ;">
```

pour un texte rouge sombre sur fond bleu clair. Une couleur de fond appliquée à du texte fait ressortir le texte en surbrillance.

La propriété `font` recouvre plusieurs propriétés apparentées :

- `font-style` définit le type de la police. Les valeurs possibles sont `normal`, `italic` et `oblique`.
- `font-weight` définit la graisse de la police. Les valeurs possibles sont les incréments successifs de 100 à 900 : `normal` (équivaut à `400`), `bold` (équivaut à `700`), `bolder` (graisse supérieure à celle assignée à une police, plus grasse que celle héritée par la police. S'il n'y en a pas, la valeur de graisse de la propriété prend la valeur numérique supérieure, l'aspect de la police ne changeant pas. Si la valeur héritée est déjà égale à `900`, la valeur résultante reste `900`), `lighter` (graisse inférieure à celle assignée à une police, moins grasse que celle héritée par la police. S'il n'y en a pas, la valeur de graisse de la propriété prend la valeur numérique inférieure, l'aspect de la police ne changeant pas. Si la valeur héritée est déjà égale à `100`, la valeur résultante devient également `100`).
- `font-size` définit la taille de la police. La valeur peut être une *taille absolue* : une *taille en points* ou un des mots-clés `xx-large`, `x-large`, `large`, `medium`, `small`, `x-small`, `xx-small`. Pour un écran, il est conseillé un facteur d'échelle de 1,2 entre les valeurs consécutives de la table. Ainsi, si la valeur `medium` correspond à `12pt`, la valeur `large` devrait correspondre à `14.4pt`. Les facteurs d'échelle peuvent varier selon les médias considérés. Par ailleurs, l'agent utilisateur devrait prendre en compte la qualité et la disponibilité des polices au moment du calcul de cette table. Celle-ci peut aussi différer d'une famille de polices à une autre. La valeur peut également être une *valeur relative*, grâce aux mots-clés `larger` et `smaller` (ce qui correspond à un déplacement d'une case dans l'échelle précédente), ou en *pourcentage*, par rapport à la taille de la police de l'élément parent.

- `font–family` modifie le type de police utilisé dans une page. Vous pouvez définir une fonte spécifique, comme `Arial` ou `Helvetica`, ou vous contenter d'utiliser un nom générique, comme `sans–serif` ou `serif`. Vous pouvez définir des familles de polices : le navigateur essaye alors chaque police à son tour et retient la première qu'il comprend. Ainsi, avec comme style

  ```
  font-family: Arial, Helvetica, sans-serif ;
  ```

  le navigateur essaye d'abord la police Arial (installée sur la plupart des PC), puis la police Helvetica (présente sur la plupart des Macintoshs) et ensuite seulement la famille générique Sans-serif, comprenant toutes les fontes dites non proportionnelles (contrairement à Arial et Helvetica, qui sont des polices proportionnelles). Parmi les autres familles génériques, vous trouvez `serif`, `cursive`, `fantasy` et `monospace`. Les familles de polices sont séparées par des virgules, mais vérifiez qu'il n'y ait pas de virgule avant le point-virgule final.



Unités de taille

Les éléments peuvent avoir recours à huit unités différentes.

- **Les unités relatives** :

`em` (taille de la fonte actuelle), `ex` (hauteur de la fonte actuelle).

- **Les unités absolues** :

`px` (pixel), `in` (pouce), `cm` (centimètre), `mm` (millimètre), `pt` (point. Un point CSS2 mesure 1/72e de pouce, soit environ 0,35 mm) et `pc` (pica, 12 points ou 4,15 mm).

Les unités absolues devraient être réservées aux cas pour lesquels vous connaissez les dimensions physiques du périphérique de sortie.

Il est souvent possible d'abréger des propriétés apparentées en les rassemblant dans une définition unique, ce qui est souvent nommé *raccourci*. C'est le cas pour la propriété `font`. Par exemple, pour spécifier qu'un élément doit se présenter dans une fonte Arial ou Sans-serif en gras italique de petite taille, vous définissez normalement comme suit les différentes propriétés :

```
{font-style: italic ; font-weight: bold ;
font-size: small ; font-family: Arial, sans-serif ; }
```

Vous pouvez cependant définir en une seule fois toutes les propriétés relatives aux polices. Vous devez pour cela définir toutes ces propriétés

selon un ordre particulier, en ne faisant apparaître que leurs valeurs. L'ordre adéquat est le suivant :

```
font-style font-weight font-size font-family
```

C'est ainsi que, dans l'exemple précédent, vous auriez pu utiliser :

```
{font: italic bold small Arial, sans-serif}
```

L'intérêt de cette technique tient à son gain de place, au prix d'une perte de lisibilité. Les propriétés individuelles sont séparées par des espaces tandis que les valeurs multiples applicables à une propriété (comme les valeurs de `font-family`) sont séparées par des virgules.

Vous découvrirez par la suite de nombreux autres exemples de notation raccourcie.

Une autre propriété de police intéressante, mais qui n'appartient pas à la notation raccourcie de la propriété `font`, est `font-variant`. Dans une police en petites capitales, les lettres minuscules présentent un aspect similaire aux lettres majuscules, avec toutefois une taille réduite et des proportions légèrement différentes. Ce type de police porte le nom de police *bicamérale* (qui possède deux casses, comme les écritures latines). Les valeurs possibles sont `normal` (une police non étiquetée comme étant en petites capitales) ou `small-caps` (qui spécifie une police étiquetée comme étant en petites capitales). En l'absence d'une telle police, les agents utilisateurs devraient en effectuer la simulation, par exemple en sélectionnant une police normale et en remplaçant les lettres minuscules par des majuscules mises à l'échelle. En dernier ressort, les lettres majuscules inchangées d'une police normale peuvent se substituer aux petites capitales : le texte apparaît alors entièrement en majuscules.

Le tableau suivant dresse la liste des principales propriétés de texte et de police.

Tableau 8.2 : *Principales propriétés de texte et de police de feuille de style CSS*

Propriété	Valeur(s)	But
`background-color`	*couleur hexadécimale ou nom*	Couleur d'arrière-plan
`color`	*couleur hexadécimale ou nom*	Couleur de texte

Tableau 8.2 : *Principales propriétés de texte et de police de feuille de style CSS*		
Propriété	**Valeur(s)**	**But**
`font`	`font-style font-weight font-size font-family`	Propriétés de police
`font-style`	`normal`, `italic`, `oblique`	Style de la police
`font-weight`	*Incréments de 100,* `normal`, `bold`, `bolder`, `lighter`	Graisse de la police
`font-size`	*taille en points,* `xx-large`, `x-large`, `large`, `medium`, `small`, `x-small`, `xx-small`, `larger`, `smaller`, *pourcentage*	Taille de la police
`font-family`	*tout nom de police,* `serif`, `sans-serif`, `cursive`, `fantasy`, `monospace`	Police(s) ou famille(s) de polices à employer
`text-align`	`left`, `right`, `center`, `justify`	Alignement du texte
`text -decoration`	`none`, `underline`, `overline`, `line-through`, `blink`	Enrichissement du texte
`text-indent`	*longueur en points, longueur en pourcentage*	Indentation du texte

Nous avions employé les propriétés `text-align`, `font-size` et `font-weight` dans plusieurs des spécifications de notre feuille de style, ainsi que `color` et `font-variant` une fois chacune :

```
/* style.css : Feuille de style pour mon site */
H1 {margin-left: 15%; margin-right: 15%; border-width: 1;
    border-style: solid; text-align: center;
    font-size: 32pt; color: blue}
H2 {font-size: 20pt; text-align: center}
P {font-size: 16pt; text-align: center}
P.legal {font-size: 8pt; font-variant: small-caps}
UL {font-size: 16 pt; text-align: center}
LI {font-size: 14pt; text-align: left; margin-left: 40%}
ADDRESS {text-align: center}
TABLE {width: 100%; border-width: 2px; border-style: double;
       text-align: center}
CAPTION {font-size: 24px; font-weight: bold;
         text-align: center;}
TH {font-size: 20pt; border-width: 1px;
    border-style: solid; text-align: center}
TD {font-size: 14pt; border-width: 1px;
    border-style: solid; text-align: center}
TD.droit {font-size: 14pt; text-align: right}
```

Représentation des éléments à l'aide d'un style

Vous connaissez déjà les éléments bloc et la façon dont ils se positionnent à l'écran. À l'aide de tableaux, vous pouvez tricher quant à cette disposition, bien que cela soit un peu délicat et demande de multiples ajustements. Jusqu'à maintenant, un conflit fondamental existait en HTML entre une bonne compatibilité internavigateurs et la possibilité d'afficher des informations sous forme graphique.

Les feuilles de style permettent d'établir un compromis entre vos exigences de présentation et les souhaits de votre public. Vous pouvez définir les éléments bloc au sein de la feuille de style dans des *propriétés de boîtes* CSS et déterminer quand et où l'élément sera affiché.

Deux catégories principales de propriétés doivent être distinguées à ce stade : celles intrinsèques à l'élément, qui définissent ses caractéristiques internes, et celles relatives à la page ou au flux du code HTML. Ces dernières définissent la position de l'élément à l'écran (ou le cas échéant sur le médium concerné).

Propriétés intrinsèques d'un élément

Tout élément CSS se comporte comme une imbrication successive de quatre boîtes :

- La boîte externe est l'aire de marges. Elle occupe le maximum de l'espace disponible (ou spécifié). Ces marges transparentes, dont les dimensions sont gérées par la propriété `margin`, définissent l'espace transparent entre le bord de l'écran (ou d'un autre élément) et l'élément actif.
- À l'intérieur de l'aire de marges se trouve l'aire de bordures. Gérée par la propriété `border`, elle représente les bordures affectées à l'élément actif.
- Vient ensuite l'aire d'espacement : la surface entre le contenu et la limite intérieure de la bordure. Elle est gérée par la propriété `padding`.
- Enfin, la boîte la plus intérieure est la boîte de contenu. Elle renferme le contenu de l'élément actif.

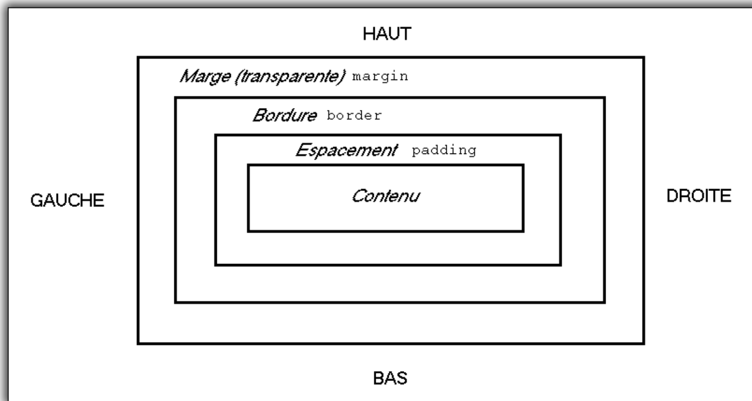


Figure 8.16 : *Boîte CSS*

La propriété `margin` et ses propriétés apparentées définissent un espace transparent entre le bord de l'écran (ou d'un autre élément) et l'élément courant. Vous pouvez définir la largeur (ou la hauteur) de chaque marge ou bien attribuer à toutes les marges une même valeur. Par exemple,

```
{margin: 10px}
```

crée une marge de 10 pixels autour de l'élément bloc. Vous pouvez spécifier séparément la taille de chaque marge en faisant suivre l'attribut d'un tiret puis de l'emplacement concerné : `margin-top`, `margin-right`, `margin-bottom` et `margin-left`. Ainsi,

```
{margin-top: 1px ; margin-right: 10px ;
 margin-bottom: 1px ; margin-left: 10px ;}
```

procure-t-il à l'élément concerné des marges supérieure et inférieure d'un pixel, mais des marges droite et gauche de 10 pixels.

Vous pouvez spécifier comme valeur pour cet attribut une *valeur absolue* (comme dans l'exemple), un *pourcentage* (par exemple, `margin: 10%` impose des marges de chaque côté de 10 % de l'espace disponible) ou choisir la valeur `auto`.

Comme pour la propriété `font`, les propriétés de boîte possèdent également une forme raccourcie, plus simple mais moins lisible. Pour la propriété `margin`, les valeurs doivent être placées dans l'ordre suivant :

```
{margin: haut droit bas gauche}
```

Remarquez que seuls des espaces séparent les valeurs individuelles. L'exemple précédent, avec une marge d'un pixel en haut et en bas et une marge de 10 pixels à droite et à gauche, pourrait ainsi être écrit :

```
{margin: 1px 10px 1px 10px}
```

Dans certains cas, des marges adjacentes peuvent fusionner. Selon la spécification CSS 2.1, les marges horizontales ne fusionnent jamais, de même que les marges verticales d'éléments positionnés de façon absolue, flottants, en ligne (`inline`) ou avec un attribut `overflow` fixé à autre chose que `visible`, ainsi que les marges de l'élément racine.

Deux ou plus marges verticales adjacentes situées dans le flux normal peuvent fusionner. La largeur de la marge résultante est la largeur maximale de la largeur des marges concernées. Dans le cas de marges négatives, la valeur absolue maximale des marges négatives est déduite de la valeur maximale des marges positives, ou de zéro en l'absence de marge positive,

Les bordures sont définies à l'aide de la propriété `border` et de ses propriétés apparentées :

- `border-width` définit la largeur de la bordure. Les largeurs peuvent être définies avec les valeurs `thin` (mince), `thick` (épaisse), `medium` (moyenne) ou exprimées en points. Vous pouvez définir individuellement la largeur de la bordure de chaque côté à l'aide de `border-top-width`, `border-right-width`, `border-bottom-width` et `border-left-width`. L'ordre abrégé de la propriété `border-width` est identique à celui de `margin`, soit :

  ```
  {border-width: haut droite bas gauche}
  ```

- `border-style` permet de définir un style de bordure. Cette propriété peut recevoir les valeurs `none` (aucune), `dotted` (pointillés), `dashed` (tirets), `solid` (pleine), `double` (grasse), `groove` (cadre gravé), `ridge` (cadre en relief), `inset` (en 3D et en creux) et `outset` (en 3D et en relief).
- `border-color` est similaire à la propriété `color` d'un élément. Elle s'emploie de la même façon.

La propriété `border` possède une forme raccourcie, qui permet d'appliquer les valeurs transmises à toutes les bordures d'un même élément. L'ordre à respecter est le suivant :

```
{border: border-width border-style color;}
```

Vous pouvez également combiner les notations raccourcies pour ne spécifier de valeur que pour la bordure du haut (ou celle de droite, du bas ou de gauche) à l'aide de la propriété `border–top` (ou `right`, `bottom` ou `left`) selon cet ordre :

```
{border-top: border-top-width border-style color;}
```

Enfin, la propriété `padding` utilise la même syntaxe que la propriété `margin`, y compris la syntaxe raccourcie. C'est une propriété établissant un espace vierge autour du contenu de l'élément, à l'intérieur de la bordure.

Les propriétés du contenu d'un élément sont `height` et `width`. Ils sont analogues aux attributs correspondants et déterminent la hauteur et la largeur du contenu. Cela signifie que, pour obtenir la taille totale de l'élément, vous devez ajouter successivement l'espacement, la bordure et la marge. Ces propriétés acceptent une valeur absolue ou en pourcentage, ainsi que la valeur `auto`. Le comportement de cette dernière valeur est quelque peu complexe et dépasse la portée de ce livre. Reportez-vous à la section 10.3, *Le calcul des largeurs et des marges*, de la spécification CSS2, à l'adresse **www.yoyodesign.org/doc/w3c/css2/visudet.html#Computing_widths_and_margins** Ces propriétés n'acceptent pas de valeurs négatives.

La propriété `overflow` permet de définir ce qui doit se passer quand le contenu de l'élément excède l'espace défini. Les valeurs acceptables sont `visible` (le contenu n'est pas tronqué et peut être représenté hors de la boîte du bloc), `hidden` (le contenu est tronqué et aucun mécanisme de défilement n'est fourni pour voir la partie rognée. La taille et la forme du reliquat du rognage peuvent être spécifiées par la propriété `clip`), `scroll` (le contenu est tronqué et l'agent utilisateur fournit si possible un mécanisme de défilement visible à l'écran, tel qu'une barre de défilement. Ce dispositif apparaît que le contenu de l'élément soit tronqué ou non) ou `auto`.

Même lorsque la valeur de la propriété `overflow` est `visible`, le contenu peut être rogné par le système d'exploitation pour tenir dans la fenêtre du document de l'agent utilisateur.

Le tableau suivant résume ces propriétés, leurs valeurs et leurs effets.

Tableau 8.3 : *Propriétés intrinsèques d'un élément*			
Propriété	**Valeur**	**Exemple**	**Effet**
`width`	*longueur, pourcentage* ou `auto`	`width: 100pt`	Impose à l'élément une largeur de cent points. Peut ainsi remplacer un tableau.
`height`	*longueur* ou `auto`	`height: 20pt`	Impose à l'élément une hauteur de vingt points
`margin`	*longueur, pourcentage* ou `auto`	`margin: 10pt`	Crée une marge de dix points autour de l'élément
`border –width`	`thin`, `medium`, `thick`, *longueur*	`border –width: thin thick thin thick`	Crée une bordure mince en haut et en bas, et une bordure épaisse sur les côtés.
`border –style`	`none`, `dotted`, `dashed`, `solid`, `double` , `groove`, `ridge`, `inset` ou `outset`	`border –style: double`	Crée une bordure épaisse autour de l'élément
`border`	`border-width border-style border-color`	`border: 4pt double #000000`	Ajoute une marge double de quatre points autour du contenu
`padding`	*longueur, pourcentage* ou `auto`	`padding: 10px`	Ajoute un espacement de dix pixels entre le contenu et la bordure

Nous avions employé plusieurs de ces propriétés dans notre fichier *style.css* :

```
/* style.css : Feuille de style pour mon site */
H1 {margin-left: 15%; margin-right: 15%;
    border-width: 1; border-style: solid;
    text-align: center; font-size: 32pt; color: blue}
H2 {font-size: 20pt; text-align: center}
P {font-size: 16pt; text-align: center}
P.legal {font-size: 8pt; font-variant: small-caps}
UL {font-size: 16 pt; text-align: center}
```

```
LI {font-size: 14pt; text-align: left; margin-left: 40%}
ADDRESS {text-align: center}
TABLE {width: 100%; border-width: 2px; border-style: double;
      text-align: center}
CAPTION {font-size: 24px; font-weight: bold;
       text-align: center;}
TH {font-size: 20pt; border-width: 1px; border-style: solid;
       text-align: center}
TD {font-size: 14pt; border-width: 1px; border-style: solid;
       text-align: center}
TD.droit {font-size: 14pt; text-align: right}
```



Incompatibilités de navigateurs anciens

Les feuilles de style ne sont pas reconnues par tous les agents utilisateurs. Par exemple, Netscape, au moins jusqu'à sa version 4.06, recensait un certain nombre de problèmes :

- La propriété `padding` des feuilles de style n'était pas reconnue. Il fallait employer à la place `topPadding`, `rightPadding`, `bottomPadding` ou `leftPadding`.
- La propriété `margins` des feuilles de style n'était pas reconnue. Il fallait employer `leftMargin`, `rightMargin` ou `bottomMargin`.
- Les propriétés `width`, `height` et `border` appliquées aux images dans les feuilles de style n'étaient pas reconnues.

Vous devez donc être conscient du fait que les utilisateurs disposant de navigateurs anciens peuvent ne pas voir votre site comme vous le souhaitez.

Propriétés de positionnement d'un élément

Ces propriétés définissent l'emplacement d'apparition de l'élément actif sur le médium considéré. Elle sont désormais reconnues par la plupart des navigateurs (Internet Explorer depuis la version 3.0 et Netscape depuis sa version 4.0).

Le meilleur moyen de les découvrir consiste à prendre un exemple. Examinez le listing suivant :

Listing 8-3 : position0.html

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
   <HEAD>
    <TITLE>"Positions CSS d'un &eacute;l&eacute;ment"</TITLE>
```

```
    <META name="author" content="votre nom">
    <META http-equiv="Content-Style-Type" content="text/css">
   </HEAD>
   <BODY style="text-align: center">
      <H1>D&eacute;monstration des
        propri&eacute;t&eacute;s de positionnement CSS
      </H1>
<!-- Flux normal -->
      <P><IMG src="Images/coucher de soleil.jpg"
        type="image/gif" width="100px"
        title="Un coucher de soleil sur le lac de Linciel">
      Premier &eacute;l&eacute;ment <CODE>P</CODE><BR>
      Ceci est un paragraphe de texte normal.
      </P>
        <P>
      <IMG src="Images/coucher de soleil.jpg"
        type="image/gif" width="100px"
        title="Un coucher de soleil sur le lac de Linciel">
      Deuxi&egrave;me &eacute;l&eacute;ment
      <CODE>P</CODE><BR>
      Ceci est un paragraphe de texte normal.
         </UL>
      </P>
      <P>
      <IMG  src="Images/coucher de soleil.jpg"
         type="image/gif" width="100px"
         title="Un coucher de soleil sur le lac de Linciel">
      Troisi&egrave;me &eacute;l&eacute;ment
      <CODE>P</CODE><BR>
      Ceci est un paragraphe de texte normal.
      </P>
      <P>
      <IMG src="Images/coucher de soleil.jpg"
        type="image/gif" width="100px"
        title="Un coucher de soleil sur le lac de Linciel">
      Quatri&egrave;me &eacute;l&eacute;ment
      <CODE>P</CODE><BR>
      Ceci est un paragraphe de texte normal.
      </P>
    </BODY>
</HTML>
```

Rien de bien excitant : la répétition de quatre paragraphes de texte, dans lesquels est insérée une image. En l'absence de spécification de position particulière, les éléments respectent le flux normal du code. En examinant ce fichier dans votre navigateur, vous observez quelque chose d'analogue à la figure suivante : un empilement des quatre paragraphes, centrés en raison de l'attribut `style="text-align: center"` de l'élément `BODY`.

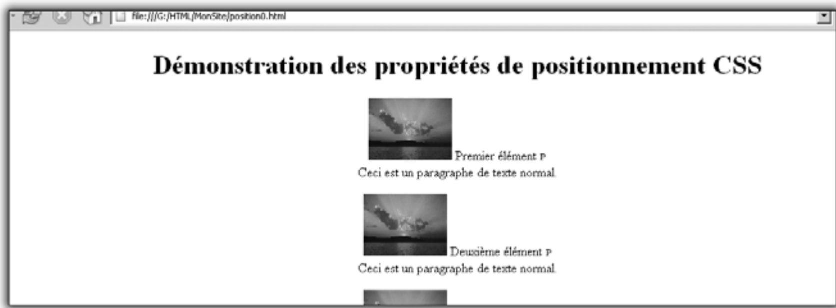


Figure 8.17 : *Flux HTML normal*

Modifions maintenant ce fichier, en ajoutant des attributs `style` aux trois derniers paragraphes, ainsi que quelques commentaires. Nous obtenons le listing suivant, dans lequel les modifications sont notées en gras :

Listing 8-4 : position.html

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
   <HEAD>
    <TITLE>"Positions CSS d'un
    &eacute;l&eacute;ment"</TITLE>
    <META name="author" content="votre nom">
    <META http-equiv="Content-Style-Type" content="text/css">
   </HEAD>
  <BODY style="text-align: center">
      <H1>D&eacute;monstration des
      propri&eacute;t&eacute;s de positionnement CSS
      </H1>
      <!-- Ce fichier utilise plusieurs éléments
similaires, dont seuls les attributs style diffèrent -->
      <!-- Flux normal -->
      <P><IMG src="Images/coucher de soleil.jpg"
        type="image/gif" width="100px"
        title="Un coucher de soleil sur le lac de Linciel">
      Premier &eacute;l&eacute;ment <CODE>P</CODE><BR>
      Flux normal.<BR>
      Ceci est un paragraphe de texte normal.
      </P>
      <!-- Image flottante -->
      <P style="float: left">
      <IMG src="Images/coucher de soleil.jpg"
        type="image/gif" width="100px"
        title="Un coucher de soleil sur le lac de Linciel">
      Deuxi&egrave;me &eacute;l&eacute;ment
```

```
        <CODE>P</CODE><BR>
        Le paragraphe est plac&eacute;e dans un cadre
        flottant &agrave; gauche.<BR>
        Ceci est un paragraphe de texte normal.
           </UL>
        </P>
        <!-- Position relative -->
        <P style="position: relative ; bottom: 10% left: 120px">
        <IMG  src="Images/coucher de soleil.jpg"
          type="image/gif" width="100px"
          title="Un coucher de soleil sur le lac de Linciel">
        Troisi&egrave;me &eacute;l&eacute;ment
        <CODE>P</CODE><BR>
        Le paragraphe est d&eacute;fini comme devant s'afficher
        en haut et &agrave; droite de l'&eacute;
        l&eacute;ment pr&eacute;c&eacute;dent.<BR>
        Ceci est un paragraphe de texte normal.
        </P>
        <!-- Position absolue -->
        <P style="position: fixed ; top: 10px ; left: 5px">
        <IMG src="Images/coucher de soleil.jpg"
         type="image/gif" width="100px"
         title="Un coucher de soleil sur le lac de Linciel">
        Quatri&egrave;me &eacute;l&eacute;ment
        <CODE>P</CODE><BR>
        Le paragraphe s'affiche de façon absolue par rapport
        à l'&agrave;cran.<BR>
        Ceci est un paragraphe de texte normal.
        </P>
    </BODY>
</HTML>
```

En examinant ce fichier dans votre navigateur, vous observez quelque chose d'analogue à la figure suivante : les quatre paragraphes sont désormais répartis à l'écran dans un ordre différent de celui de leur apparition dans le code.

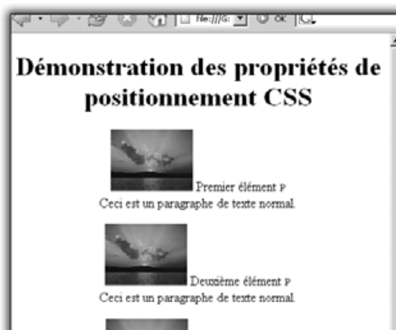


Figure 8.18 : *Positionnements d'éléments*

Analysons un peu ce qui se passe :

- Le premier paragraphe respecte le flux HTML normal. Il est entré en dessous du titre, exactement comme dans la copie d'écran précédente.
- Le second paragraphe est placé dans une boîte flottante positionnée à gauche : il est retiré du flux normal.
- Le troisième paragraphe se voit attribuer une position relative par rapport à l'élément précédent : légèrement plus haut et franchement plus à droite.
- En revanche, le quatrième paragraphe se voit attribuer une position absolue par rapport à l'écran : en haut et à gauche de celui-ci.

Le dernier paragraphe est particulièrement intéressant. Pour vous en convaincre, réduisez la largeur de la fenêtre de votre navigateur, et faites, si nécessaire et possible, défiler la fenêtre : le quatrième paragraphe se déplace en conséquence et occupe toujours le même emplacement à l'écran !



Figure 8.19 : *Un élément positionné de façon absolue occupe toujours la même position à l'écran*

Cet exemple devrait vous ouvrir un certain nombre de possibilités. Tout d'abord, plus besoin de tableaux, puisque vous pouvez définir l'emplacement d'un élément en fonction de l'élément précédent. Ensuite, un élément défini en position absolue (par exemple en haut à gauche) peut constituer un magnifique logo, toujours affiché au même emplacement, même lorsque l'utilisateur fait défiler une page…

Remarquez toutefois que ces types de positionnement peuvent amener un « recouvrement » des différents éléments, ce qui n'est guère

souhaitable. Heureusement, comme vous allez le voir, il existe des moyens de pallier ce problème…

Regardons de plus près les différentes propriétés mises en œuvre lors du positionnement d'éléments.

La propriété `float` spécifie qu'une boîte doit être flottante à gauche, à droite ou pas du tout. Elle peut être spécifiée pour tout élément, mais ne s'applique qu'aux éléments qui ne sont pas positionnés de façon absolue.

La propriété `clear` (non employée dans cet exemple) indique quels côtés d'une boîte d'élément ne peuvent pas être adjacents à une boîte flottante antérieure. Elle ne concerne pas les éventuels éléments flottants à l'intérieur de l'élément concerné ou dans d'autres contextes de mise en forme. C'est un espacement au-dessus de la marge supérieure d'un élément, qui sert à « pousser » verticalement l'élément au-delà de la boîte flottante, généralement vers le bas.

La propriété `position` détermine si la position d'un élément est `absolute`, `relative` ou `static`. Sa valeur par défaut est `static`, ce qui signifie que l'élément apparaît selon son chargement, en principe selon sa place dans le code source. Une position `absolute` signifie que la position de l'élément est fixe par rapport au coin supérieur gauche du médium (généralement l'écran). Comme nous l'avons signalé, un élément doté de ce type de position peut recouvrir un autre élément, puisqu'il est retiré du flux normal.

Si la propriété `position` possède comme valeur `relative`, l'élément sera placé en fonction de son parent ou de son voisin. Si l'élément parent a recours à une position `absolute`, alors l'élément `relative` apparaîtra à la suite de celui-ci et pourra également recouvrir d'autres éléments : il est lui aussi retiré du flux normal.

Les propriétés `left` et `top` définissent l'endroit où l'élément doit être placé par rapport aux côtés de l'écran (ou, dans le cas d'un élément enfant, aux côtés de l'élément parent). Les valeurs sont exprimées en points ou en pourcentage.

Pour gérer les possibles recouvrements, servez-vous de la propriété `z-index`. Elle permet de contrôler l'élément devant apparaître au-dessus d'un autre en cas de recouvrement de positions absolues : un peu comme les couches d'un programme de dessin.

clip définit un rognage de l'élément, tandis que visibility permet d'établir si l'élément est visible ou non. Un élément visibility: hidden occupe son espace, mais n'est pas visible à l'écran.

Ces propriétés, leurs valeurs et leurs buts sont présentés dans le tableau suivant.

Tableau 8.4 : *Propriétés CSS de positionnement des éléments*

Propriété	Valeur	Exemple	Effet
float	left, right, none ou inherit	float: left	Place l'élément dans un rectangle flottant sur la gauche
position	absolute, relative, static	position: absolute	Place l'élément à l'écran sans tenir compte des autres éléments de la page
top	*longueur ou pourcentage*	top: 10pt	Place l'élément à dix points du haut du document ou de l'élément parent
left	*longueur ou pourcentage*	left: 15pt	Place l'élément à quinze point de la gauche du document ou de l'élément parent
clip	shape (top-value, right-value, bottom-value, left-value)	clip: rect (5 10 5 10)	Rogne cinq pixels du haut et du bas, et dix pixels sur les côtés.
overflow	visible, hidden, auto, scroll	overflow: scroll	Attribue à l'élément une barre de défilement, qu'il en ait besoin ou non.
z-index	*Entier*	z-index: 1	Place d'abord le premier élément et empile dessus les suivants
clear	none, left, right, both ou inherit	Clear: left	La boîte est décalée de l'espace nécessaire pour placer le bord de la bordure supérieure en dessous du bord inférieur de toute boîte flottante due à un élément antérieur dans le document source.

Une mise en œuvre efficace de ces propriétés demande souvent pas mal de tâtonnements. Le résultat est toutefois souvent à la hauteur des efforts engagés… Souvenez-vous que nous avions dit que les feuilles de style représentaient un aspect avancé de HTML : vous en avez ici une preuve.

8.6. Feuille de style en cascade

Les langages de feuille de style en cascade, tel que CSS, autorisent l'assemblage des informations de style provenant de plusieurs sources. Pour définir une cascade, l'auteur spécifie une succession d'éléments `LINK` et/ou `STYLE`. Les informations de style se répandent en cascade selon l'ordre d'apparition des éléments dans la section `HEAD`.



Gare aux mélanges…
La spécification HTML 4.01 ne précise pas la manière dont cascadent les feuilles de style issues de différents langages de style. Mieux vaut éviter de mélanger les langages de feuilles de style.

La cascade peut inclure des feuilles de style qui concernent différents médias. Les éléments `LINK` et `STYLE` peuvent tous deux être utilisés avec l'attribut `media`. L'agent utilisateur est alors responsable de l'élimination des feuilles de style qui ne s'appliquent pas au médium courant.

Cette notion de « cascade » semble alors poser quelques problèmes : comment l'agent utilisateur sait-il quelle feuille de style doit être appliquée ? Pour le cas d'un médium de sortie, la chose est claire, mais nous avons vu qu'il pouvait exister plusieurs feuilles de style en cascade (une ou plusieurs feuilles externes, des éléments `STYLE` et des attributs `style`). Cela nous amène à la notion d'*héritage*.

Cascade et héritage

Lorsqu'un agent utilisateur veut restituer un document, il doit identifier les valeurs des propriétés de style : la famille de police, le style de la police, la taille, la hauteur de ligne, la couleur du texte, etc. Le mécanisme exact dépend du langage de feuille de style, mais la description qui suit est généralement applicable.

Le mécanisme de cascade est utilisé quand un certain nombre de règles de style s'appliquent toutes directement à un élément. Il permet à l'agent utilisateur de classer les règles selon leur spécificité pour déterminer celle qui est applicable. Si aucune règle n'est trouvée, l'étape suivante va dépendre de la propriété de style : est-elle héritable ? Toutes ne le sont pas. Le langage de feuille de style fournit alors des valeurs par défaut, utilisées en l'absence de règle explicite pour un élément particulier.

Si la propriété peut s'hériter, l'agent utilisateur examine l'élément englobant immédiat pour voir si une règle s'y applique. Ce processus continue jusqu'à l'identification d'une règle applicable. Ce mécanisme autorise une spécification compacte des feuilles de style. Par exemple, l'auteur peut spécifier le type d'alignement du texte pour tous les éléments de `BODY` par une seule règle appliquée à l'élément `BODY`. C'est ce que nous avions fait dans les fichiers *position0.html* et *position.html* :

```
<BODY style="text-align: center">
```

Fondamentalement, tout élément inclus dans un autre hérite du style de celui-ci, à moins que la feuille de style ne précise le contraire. Par exemple, la plupart des éléments sont inclus dans l'élément `BODY`. Si celui-ci est défini comme devant comporter du texte centré de couleur bleue, tous les éléments de `BODY` possèderont un texte bleu centré. Même si un élément particulier, comme `H`, est également défini dans la feuille de style, il hérite des propriétés de `BODY` à moins que ses propriétés `color` et `background` ne soient explicitement spécifiées dans la feuille de style.

La théorie de l'héritage est donc très simple : vous partez de l'information la plus externe (la feuille de style externe) et progressez vers l'intérieur. Un élément inclus dans un autre élément hérite de ses propriétés sauf s'il dispose d'un style modifiant tout ou partie de ces propriétés. Une feuille de style externe est surpassée par une feuille interne, elle-même dépassée par un style défini à l'intérieur de l'élément, lui-même surpassé pour un élément enfant par un style défini dans celui-ci.

Voici l'ordre d'héritage, par priorité croissante :

- Propriétés définies par l'utilisateur pour l'élément courant.
- Propriétés de feuille de style externe de l'élément parent.
- Propriétés de feuille de style locale de l'élément parent.

- Styles de ligne de l'élément parent.
- Propriétés de feuille de style externe de l'élément courant.
- Propriétés de feuille de style locale de l'élément courant.
- Styles de ligne de l'élément courant.

Vous voyez apparaître dans cette liste un élément « curieux » : la notion de feuille de style définie par l'utilisateur. Vous l'avez probablement déjà fait sans le savoir : il s'agit de la modification des propriétés de votre agent utilisateur, par exemple de votre navigateur.

Prenons un exemple. Avec Firefox d'abord :

1 Ouvrez Firefox, puis chargez le fichier *index.html* de votre site. Vous devez voir votre familière page d'accueil.



Figure 8.20 : *Page d'accueil de votre site*

2 Choisissez dans la barre de menu **Affichage** > **Style de la page** > **Aucun style**. Toute la mise en forme définie dans la feuille de style disparaît.



Figure 8.21 : *Page d'accueil de votre site, sans feuille de style.*

Choisissez à nouveau **Affichage** > **Style de la page** puis sélectionnez **Style de base de la page** pour retrouver un affichage normal.

Vous avez ici ordonné à l'agent utilisateur d'ignorer toute feuille de style. Vous pouvez toutefois agir sur la page d'autres façons.

3 Choisissez dans la barre de menu **Affichage** > **Taille de texte** > **Plus grande**. Tout le contenu texte de la page voit sa taille de police augmentée.



Figure 8.22 :
Page d'accueil de votre site, avec augmentation de la taille des polices.

L'effet est exactement identique à l'application générale de la propriété `font-size: larger`. Vous pouvez augmenter à plusieurs reprises la taille des polices, ou inversement la diminuer. Après avoir procédé à quelques essais, choisissez **Affichage** > **Taille de texte** > **Normale** pour retrouver un affichage normal.

Vous pouvez également agir sur la police par défaut, ainsi que sur la couleur de divers éléments, en choisissant **Outils** > **Options**. Sélectionnez **Général** et choisissez **Couleurs et police**.



Figure 8.23 :
Fenêtre couleurs et polices de Firefox

D'autres paramètres peuvent également être modifiés dans la section *Avancé* des **Options**, comme le redimensionnement automatique des images trop grandes pour l'affichage dans la fenêtre de navigation.

Des modifications similaires sont possibles avec Internet Explorer (à partir de sa version 4) et celui-ci offre en outre la possibilité d'employer une feuille de style personnalisée.

1 Ouvrez Internet Explorer, puis cliquez sur **Outils** > **Options Internet**.

2 Dans l'onglet **Général**, cliquez sur le bouton **Accessibilité**.

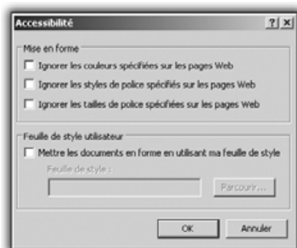


Figure 8.24 :
Fenêtre Accessibilité d'Internet Explorer

3 Cochez la case à côté de l'option *Mettre les documents en forme en utilisant ma feuille de style*, puis saisissez dans la boîte de dialogue le chemin d'accès et le nom de la feuille.

4 Remarquez que, même sans définir de feuille de style personnelle, vous pouvez décider dans cette fenêtre d'ignorer les spécifications de couleur, de nom et de taille de polices des pages que vous affichez.

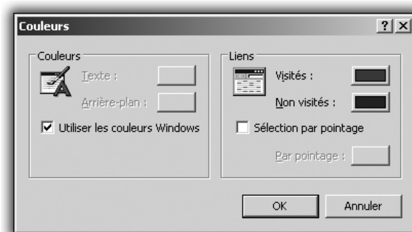


Figure 8.25 :
Fenêtre permettant de définir vos préférences, qui outrepassent les spécifications de style des pages affichées.

5 Cliquez sur OK pour sortir de l'option *Accessibilité*. Les autres boutons de la fenêtre **Général** permettent de spécifier les couleurs, les polices et la langue.



Figure 8.26 : *Autre fenêtre.*

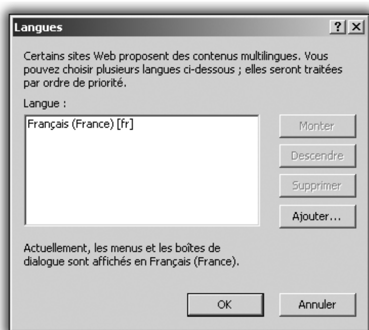


Figure 8.27 : *Encore une autre.*

Cliquez de nouveau sur OK pour enregistrer vos nouveaux réglages des **Options Internet**.

L'écran affiche alors les pages en mêlant les feuilles de style de l'utilisateur et celles du créateur du site.

8.7. Feuille de style auditive, pour un public particulier

Comme nous l'avons signalé, vous pouvez définir plusieurs feuilles de style, adaptées chacune aux caractéristiques du médium sur lequel le document doit être restitué (par exemple : écran graphique, écran de télévision, appareil de poche, navigateur à synthèse vocale, appareil tactile braille, etc.).

Les feuilles de style disposent de plusieurs moyens pour présenter des informations à des utilisateurs parcourant le Web à l'aide de navigateurs non standard. Par exemple, certaines propriétés des feuilles de style

permettent de définir la pagination, en cas d'impression de la page. De façon similaire, le W3C a configuré plusieurs propriétés concernant des périphériques auditifs ou audio.

Une feuille de style auditive peut être utilisée par un malvoyant, avec des navigateurs mains libres ou des périphériques comparables texte-parole. Outre permettre la présentation du contenu normal d'une page Web, une feuille de style auditive autorise les concepteurs Web à avoir recours à des effets sonores particuliers, comme des précisions, des orateurs multiples ou des modifications d'intonation.

Voici un exemple de feuille de style possédant des propriétés auditives :

Listing 8-5 : styleaudit.css

```
/* styleaudit.css : Feuille de style à propriétés auditives */
BODY {color: #000000 ; background: #FFFFFF; text-align: left}
.direction {margin-left: 10%; font-family: monospace;
   font-style: italic; color: #000080; speak: none}
SPAN.direction {margin-left: 0px;}
.Capulet {speak: normal; voice-family: male}
.Lady_capulet {speak: normal; voice-family: female;
   speech-rate: slow; richness: 40%; pitch: hight}
.Montague {speak: normal; voice-family: male; richness: 80%;
   pitch: low}
.Lady_montague {speak: normal; voice-family: female;
   speech-rate: slow}
.off {font-variant: small-caps; speak: normal;
   voice-family: male; volume: 75%}
```

Remarquez l'emploi de classes (`.direction`, par exemple) sans signalisation par un élément : le style défini s'applique alors à toutes les instances de cette classe, quel que soit l'élément concerné. Comme nous voulons supprimer l'indentation pour une indication de mise en scène en ligne, nous avons spécifié une marge à `0` pour l'élément `SPAN.direction`.

Cette feuille de style est liée au fichier suivant :

Listing 8-6 : romeo.html

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>"Hamlet de Shakespeare"</TITLE>
<META name="author" content="Fabrice Lemainque">
<META name="version" content="1.8.1">
```

```
<META http-equiv="Content-Style-Type" content="text/css">
<LINK rel="STYLESHEET" type="text/css" href="styleaudit.css">
</HEAD>
<BODY>
<H1 align="center">Extrait de Rom&eacute;o et Juliette
  de Shakespeare</H1>
<HR>
<DIV>
<SPAN class="off">Capulet. - </SPAN>
<SPAN class="Capulet">Quel est ce bruit ? ...
   Hol&agrave; ! qu'on me donne ma grande
   &eacute;p&eacute;e.</SPAN><BR>
<SPAN class="off">Lady Capulet. - </SPAN>
<SPAN class="Lady_capulet">Non, une b&eacute;quille !
   une b&eacute;quille !.. Pourquoi demander une
    &eacute;p&eacute;e ?</SPAN><BR>
<SPAN class="off">Capulet. - </SPAN>
<SPAN class="Capulet">Mon &eacute;p&eacute;e dis-je !
   le vieux Montague arrive et brandit sa rapi&egrave;re
   en me narguant !</SPAN><BR>
</DIV>
<P class="direction">Entrent Montague, l'&eacute;p&eacute;e
  à la main, et Lady Montague.</P>
<DIV>
<SPAN class="off">Montague. - </SPAN>
<SPAN class="Montague">
  A toi, mis&eacute;rable Capulet ! Ne me retenez pas !
  L&acirc;chez-moi !</SPAN><BR>
<SPAN class="off">Lady Montague</SPAN>
<SPAN class="direction">
  , le retenant</SPAN>
<SPAN class="off">. - </SPAN>
<SPAN class="Lady_montague">Tu ne feras pas un seul pas vers
  ton ennemi.</SPAN>
</DIV>
<P class="direction">Entre le Prince Escalus, avec sa suite.
</P>
<HR>
<CITE>Rom&eacute;o et Juliette, Acte I Sc&agrave;ne I.
William Shakespeare,"Richard III, Rom&eacute;o et Juliette,
Hamlet", Trad. F-V Hugo. 1979, Ed. Garnier-Flammarion,
ISBN 2-08-070006-5, page 154.
</CITE>
<HR>
</BODY>
</HTML>
```

Affiché dans un navigateur, celui-ci offre un aspect tout à fait normal.

Avec un agent utilisateur texte-parole, les tirades sont prononcées à l'aide des voix adéquates. Si toute la pièce était ainsi mise en forme

dans un fichier HTML, il serait possible à tous les acteurs de répéter à l'aide d'un ordinateur : il suffirait de désactiver la lecture des répliques qui le concerne dans la feuille de style. Les autres répliques seraient encore lues par l'agent utilisateur.

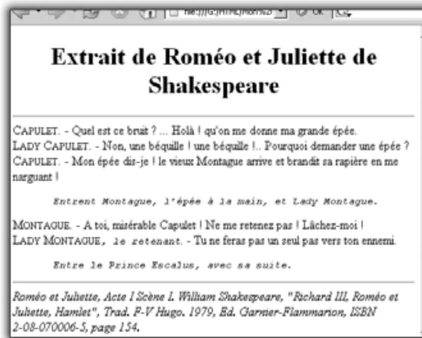


Figure 8.28 :
Aspect de l'extrait de Roméo et Juliette

La propriété sonore la plus importante est `speak`. Pour désactiver totalement la lecture vocale d'un élément, fixez sa valeur à `none`, comme nous l'avons fait pour les indications de mise en scène :

```
.direction {margin-left: 10%; font-family: monospace;
font-style: italic; color: #000080; speak: none}
```

Cet élément ne sera pas lu (parlé), bien que cette propriété puisse être ignorée par certains convertisseurs texte-parole. La valeur `spell-out` permet aux mots concernés d'être épelés, la valeur `normal` correspondant à un rendu vocal normal.

Dans de nombreux navigateurs texte-parole, les liens hypertextes sont lus comme du texte ordinaire. Pour donner un indice sonore particulier lorsque le navigateur atteint un lien hypertexte, servez-vous d'un signal sonore grâce à la propriété `cue`. Par exemple :

```
cue: url("pop.au")
```

Vous pouvez placer l'indice avant ou après, à l'aide des propriétés `cue-before` et `cue-after`.

Vous pouvez aussi facilement ajouter une courte pause avant ou après le signal. Utilisez ainsi

```
pause: 20ms
```

pour créer une pause de vingt millisecondes avant et après l'élément. Vous pouvez également avoir recours aux propriétés `pause-before` et

`pause-after` avec la même syntaxe. La propriété `pause` dispose d'un raccourci :

```
pause: pause-before-value pause-after-value
```

Comme cela était mentionné plus haut, une feuille de style auditive permet d'avoir recours à plusieurs orateurs. Les propriétés `azimuth` (direction), `elevation` (hauteur) et `voice-family` permettent d'ajouter des effets saisissants à votre page Web. Bien évidemment, ces propriétés peuvent être ou non rendues correctement selon l'agent utilisateur concerné, mais cette démarche ambitieuse est la première étape vers un format Web audio utilisable de façon universelle.

Servez-vous de la propriété `volume` pour ajuster le niveau sonore rendu par le programme. Une valeur `silent` impose au programme d'attendre aussi longtemps qu'il l'aurait fallu pour lire le texte, mais sans que vous n'entendiez rien : c'est ce qu'il faudrait employer pour faire répéter l'acteur jouant Capulet, en modifiant comme suit son style :

```
.Capulet {speak: normal; voice-family: male; volume: silent}
```

Voici quelques autres moyens de modifier les propriétés de la voix : `pitch` rend la voix plus aiguë ou plus grave, `voice-family` permet de choisir entre une voix d'homme, de femme ou d'enfant, ou bien d'utiliser une « police de voix » propre au navigateur. `speech-rate` accélère ou ralentit le débit vocal, tandis que `richness` et `stress` autorisent la modification de certains attributs de la voix elle-même.

Nous avons ici employé une unique feuille de style qui combinait présentations visuelle et sonore. Il existe deux autres méthodes pour mieux séparer les propriétés en fonction du périphérique de sortie. N'oubliez pas que, chaque fois que vous répartissez des instructions en catégories bien distinctes, vous facilitez d'autant la maintenance ultérieure. La première méthode consiste à employer au sein d'un unique fichier la règle `@media` :

Listing 8-7 : styleaudit2.css

```
/* styleaudit2.css : Feuille de style à propriétés visuelles
 et auditives */
@media screen(
    BODY {color: #000000 ; background: #FFFFFF; text-align: left}
    .direction {margin-left: 10%; font-family: monospace;
       font-style: italic; color: #000080}
    SPAN.direction {margin-left: 0px;}
    .off {font-variant: small-caps}
    )
@media speech (
```

```
    .direction {speak: none}
    .Capulet {speak: normal; voice-family: male}
    .Lady_capulet {speak: normal; voice-family: female;
      speech-rate: slow; richness: 40%; pitch: hight}
    .Montague {speak: normal; voice-family: male;
      richness: 80%; pitch: low}
    .Lady_montague {speak: normal; voice-family: female;
      speech-rate: slow}
     .off {speak: normal; voice-family: male; volume: 75%}
     )
```

Il aurait également été possible de créer deux feuilles distinctes et de recourir à un attribut `media` lors de la liaison de ces feuilles au fichier concerné :

Listing 8-8 : visuel.css

```
/* visuel.css : Feuille de style à propriétés uniquement
visuelles */
BODY {color: #000000 ; background: #FFFFFF; text-align: left}
.direction {margin-left: 10%; font-family: monospace;
  font-style: italic; color: #000080}
SPAN.direction {margin-left: 0px;}
.off {font-variant: small-caps}
```

Listing 8-9 : audit.css

```
/* audit.css : Feuille de style à propriétés purement
auditives */
.direction {speak: none}
.Capulet {speak: normal; voice-family: male}
.Lady_capulet {speak: normal; voice-family: female;
  speech-rate: slow; richness: 40%; pitch: hight}
.Montague {speak: normal; voice-family: male;
  richness: 80%; pitch: low}
.Lady_montague {speak: normal; voice-family: female;
  speech-rate: slow}
.off {speak: normal; voice-family: male; volume: 75%}
```

Vous liez alors ces deux feuilles de style au fichier *romeo.html* à l'aide des deux éléments `LINK` suivants, dotés d'un attribut `media` :

```
<LINK rel="STYLESHEET" type="text/css" href="visuel.css"
  media="screen">
<LINK rel="STYLESHEET" type="text/css" href="audit.css"
   media="speech">
```

Nous avons examiné ici le cas d'une présentation à la fois visuelle et auditive, mais vous pourriez faire de même pour une présentation en vue d'une impression, ou pour distinguer une présentation sur écran d'ordinateur d'une présentation à l'aide d'un vidéo-projecteur.



Pourquoi des différences entre impression, écran d'ordinateur et vidéo-projection ?

Un écran d'ordinateur fonctionne par émission de lumière : celle-ci provient soit de son tube cathodique, soit des cristaux liquides ou d'un dispositif similaire. De ce fait, des pages plutôt claires avec un texte sombre sont généralement préférables. Ce fait est encore plus flagrant pour une impression, surtout en noir et blanc, pour laquelle il vaut mieux réduire les surfaces sombres imprimées (pour des raisons d'économies d'encre et de lisibilité).

C'est exactement le contraire avec un vidéo-projecteur ou un rétroprojecteur, puisque les participants ne voient qu'une lumière réfléchie par l'écran : des pages foncées avec un texte clair sont généralement moins fatigantes et possèdent un impact supérieur. Inconvénient toutefois pour des transparents de rétroprojection : vous consommez alors beaucoup plus d'encre…

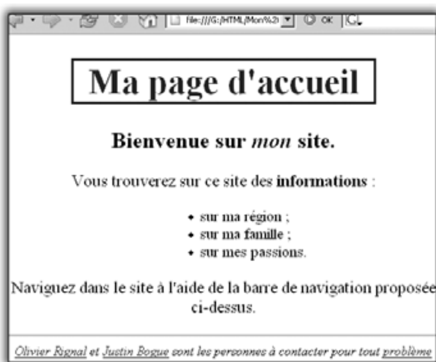


Figure 8.29 : *Une telle page est parfaite pour un affichage écran ou une impression*

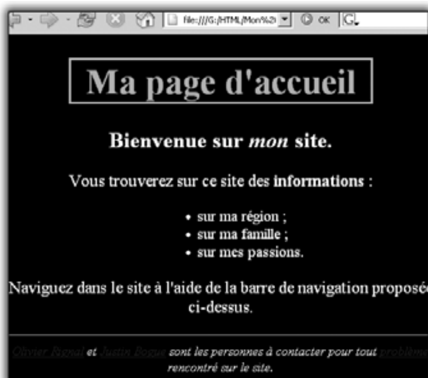


Figure 8.30 : *En revanche, cette version est mieux adaptée à une vidéo-projection.*

En conclusion, les feuilles de style vous offrent un mécanisme d'une rare puissance pour adapter parfaitement le contenu de vos pages et documents Web au public visé. Leur étude complète dépasse largement la portée de ce livre, puisque la spécification CSS2 est aussi, sinon plus complexe que la spécification HTML 4.01.

8.8. Résumé

- La plupart des attributs et éléments relatifs à la mise en forme et à la présentation des pages Web sont désormais déconseillés en faveur du recours aux feuilles de style.
- La spécification HTML 4.01 ne lie pas HTML à un langage de style particulier. Cela permet l'utilisation potentielle de plusieurs langages. Le langage le plus fréquemment employé est CSS (*Cascading Style Sheet*), fondé sur la Recommandation W3C CSS 2.1 (**http://www.w3.org/TR/CSS21/cover.html#minitoc**), dans l'attente de la diffusion finale de la version 3.
- Vous spécifiez le langage de feuille de style à l'aide d'un élément `META`.
- Vous pouvez inclure des styles dans une page Web de trois façons différentes :
  - à l'aide de l'attribut `style`, placé dans pratiquement dans n'importe quel élément ;
  - à l'aide d'un élément `STYLE` placé dans l'élément `HEAD`, un élément `STYLE` pouvant renfermer plusieurs définitions de style ;
  - en créant une feuille de style externe, indépendante du document HTML. Vous liez ensuite cette feuille externe à votre document HTML à l'aide d'un élément `LINK`.
- Vous pouvez définir le style de toutes les occurrences d'un élément, d'une classe spécifique ou d'un `id` spécifique. Ces possibilités ne sont pas mutuellement exclusives.
- HTML permet aux auteurs de spécifier les documents exploitant les caractéristiques du médium sur lequel le document doit être restitué (par exemple : écran graphique, écran de télévision, appareil de poche, navigateur à synthèse vocale, appareil tactile braille, etc.). Cela est effectué à l'aide de l'attribut `media` de l'élément `STYLE`.

- Le contenu de l'élément `STYLE` doit être masqué vis-à-vis des agents utilisateurs qui ne reconnaissent pas les feuilles de style.
- Une feuille de style est un simple fichier texte qui indique à l'agent utilisateur comment restituer votre code HTML. Les feuilles de style externes emmènent HTML un pas plus avant dans la séparation réussie des logiques de travail et de présentation. Un nom de fichier de feuille de style se termine par une extension *.css*.
- Une feuille de style externe peut être définie comme persistante, préférée ou alternative.
- Outre les propriétés de modification des caractéristiques du texte et des polices, CSS propose des propriétés de positionnement. Celles-ci permettent de s'affranchir en partie ou en totalité du flux normal du code HTML.
- Tout élément CSS se comporte comme une boîte dotée de propriétés soit intrinsèques, soit relatives à un ou plusieurs autres éléments ou au document.
- Les langages de feuille de style en cascade, tel que CSS, autorisent l'assemblage des informations de style provenant de plusieurs sources. Pour définir une cascade, l'auteur spécifie une succession d'éléments `LINK` et/ou `STYLE`. Les informations de style se répandent en cascade selon l'ordre d'apparition des éléments dans la section `HEAD`.
- Un élément inclus dans un autre élément hérite de ses propriétés sauf s'il dispose d'un style modifiant tout ou partie de ces propriétés. Une feuille de style externe est surpassée par une feuille interne, elle-même dépassée par un style défini à l'intérieur de l'élément, lui-même surpassé pour un élément enfant par un style défini dans celui-ci.
- Il est possible de paramétrer un agent utilisateur afin qu'il ignore tout ou partie des feuilles de style prévues par l'auteur, et même parfois d'appliquer sa propre feuille de style utilisateur.
- Une feuille de style à propriétés auditives est un exemple de feuille de style adaptée aux caractéristiques du médium sur lequel le document doit être restitué.

Chapitre 9

Formulaires

Constituants d'un formulaire 339
Formulaire et focus 362
Commandes inactives et en lecture seule 368
Soumission du formulaire 370
Remarques à propos des formulaires 375
Résumé 376

Un formulaire constitue un moyen d'apporter un peu d'interactivité à une page HTML. Il peut permettre de sélectionner une langue, une feuille de style particulière, de personnaliser l'affichage (comme pour un « Bienvenue *votre_nom* ») et bien d'autres choses encore.

Dès le prochain chapitre, quand vous en apprendrez plus sur les scripts, les formulaires deviendront très importants. La plupart des scripts ont recours à des formulaires pour ajouter du dynamisme à un site. Un script peut valider le contenu d'un formulaire avant de le soumettre à un autre script, ou même traiter la totalité du contenu sans recourir aux ressources du serveur. Certains de ces scripts peuvent enregistrer des informations dans un ou plusieurs cookies qui sont enregistrés sur l'ordinateur de l'utilisateur, et non sur le serveur Web. Grâce aux scripts et aux formulaires, vous pouvez créer de puissantes applications qui réduisent le poids imposé aux ressources de votre serveur.

Un formulaire HTML est une partie du document constituée d'un contenu normal, d'un balisage, mais surtout d'éléments spéciaux étiquetés appelés *commandes* (cases à cocher, boutons d'option, menus, etc.). L'utilisateur « remplit » généralement le formulaire en modifiant ses commandes (en saisissant un texte, en sélectionnant les articles d'un menu, etc.), avant de le soumettre à un agent pour traitement (par exemple, à un serveur Web, à un serveur de messagerie, etc.).

Ce chapitre présente les principaux constituants des formulaires Web, ainsi que la façon de les concevoir. Le Chapitre 10, consacré aux scripts, traitera plus en détail de l'emploi de scripts pour traiter les informations d'un formulaire.

Éléments et attributs étudiés :

```
FORM, action, method, name, id, enctype, accept-charset,
  accept
LABEL, for
INPUT, type, checked, maxlength, notab, size, src,
  tabindex, value
ISINDEX
BUTTON, name, value, type
SELECT, name, size, multiple, OPTION, label, value,
  selected, OPTGROUP, label
TEXTAREA, name, rows, cols
LABEL, value, for
FIELSET, LEGEND, align
  tabindex, accesskey, disabled, readonly
OBJECT
```

9.1. Constituants d’un formulaire

Élément FORM

Voici un extrait de fichier qui crée un formulaire simple. Celui-ci comprend des étiquettes ou libellés (*labels*), des boutons d’option et des boutons de commande (pour réinitialiser le formulaire ou le soumettre) :

```
<FORM action="http://unsite.com/prog/ajoutermembre"
  method="post">
  <P>
    <LABEL for="prenom">Pr&eacute;nom : </LABEL>
    <INPUT type="text" id="prenom"><BR>
    <LABEL for="nom">Nom : </LABEL>
    <INPUT type="text" id="nom"><BR>
    <LABEL for="email">e-mail : </LABEL>
    <INPUT type="text" id="email"><BR>
    <INPUT type="radio" name="genre" value="homme"> Homme<BR>
    <INPUT type="radio" name="genre" value="femme"> Femme<BR>
    <INPUT type="submit" value="Envoyer"> <INPUT type="reset">
  </P>
 </FORM>
```

Affiché dans un navigateur, cet extrait de fichier affiche l’aspect présenté dans la figure suivante.

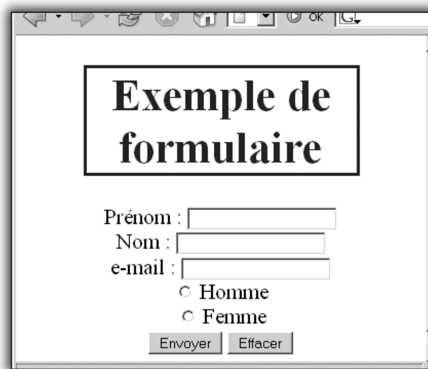


Figure 9.1 :
Exemple de formulaire

Il est vrai que l’aspect n’est pas parfait, mais vous savez comment l’améliorer si vous le voulez…

Vous créez un formulaire dans une page HTML à l’aide de l’élément `FORM`. Ses balises d’ouverture et de fermeture sont obligatoires.

L'élément `FORM` agit comme conteneur général du formulaire, notamment pour les commandes. Il spécifie :

- La disposition du formulaire (définie par le contenu de l'élément).
- Le programme qui va manipuler le formulaire rempli puis soumis. Le programme récepteur doit être capable d'analyser les couples nom/valeur afin de les utiliser.
- La méthode selon laquelle les données de l'utilisateur seront envoyées au serveur.
- L'encodage de caractères qui doit être accepté par le serveur pour manipuler ce formulaire. Les agents utilisateurs peuvent avertir l'utilisateur de l'encodage accepté et/ou empêcher que celui-ci n'entre des caractères non reconnus.

Un formulaire peut contenir un texte et un balisage (paragraphes, listes, etc.) en plus de commandes. Il accomplit tout ceci à l'aide de plusieurs attributs fondamentaux :

- L'attribut `action` spécifie l'agent chargé du traitement du formulaire. Le comportement de l'agent utilisateur pour une valeur autre qu'un URI http est indéfini.
- L'attribut `method` spécifie la méthode HTTP qui sera employée pour soumettre le jeu des données de formulaire. Les valeurs possibles (insensibles à la casse) sont `get` (la valeur par défaut) et `post`. Nous reviendrons plus loin sur la signification de ces deux méthodes.
- L'attribut `name` nomme l'élément pour qu'il puisse être appelé par une feuille de style ou un script. Cet attribut est désormais déconseillé : mieux vaut employer l'attribut `id` pour identifier les éléments.
- L'attribut `enctype` spécifie le type de contenu défini pour la soumission du formulaire au serveur (lorsque la valeur de `method` est `post`). La valeur par défaut de cet attribut est `application/x-www-form-urlencoded`. La valeur `multipart/form-data` doit être employée avec l'élément `INPUT` lorsque celui-ci est défini comme `type="file"`.
- L'attribut `accept-charset` spécifie la liste des encodages de caractères des données saisies qui sont acceptés par le serveur traitant ce formulaire. La valeur est une liste de valeurs de jeu de caractères, séparées par des espaces et/ou des virgules. Le client doit interpréter cette liste comme une liste de type « OU

exclusif ». La valeur par défaut de cet attribut est la chaîne réservée `UNKNOWN`. Les agents utilisateurs peuvent interpréter cette valeur comme représentant l'encodage de caractères employé pour transmettre le document contenant l'élément `FORM` en question.

- Enfin, l'attribut `accept` spécifie la liste de types de contenu, séparés par des virgules, que le serveur qui traite ce formulaire prend correctement en charge. L'agent utilisateur peut utiliser ces informations pour éliminer les fichiers non conformes quand il demande à l'utilisateur de sélectionner un fichier à envoyer au serveur.

Types de commandes de formulaire

Les utilisateurs interagissent avec les formulaires au moyen de commandes nommées. Le « nom de commande » d'une commande est spécifié par son attribut `name`. La portée de l'attribut `name` d'une commande au sein d'un élément `FORM` est cet élément `FORM`.

Chaque commande possède à la fois une *valeur initiale* et une *valeur courante*, toutes deux des chaînes de caractères. La *valeur initiale* d'une commande peut généralement être spécifiée à l'aide de l'attribut `value` de l'élément de commande.

La *valeur courante* d'une commande est d'abord égale à la valeur initiale. Elle peut être modifiée par la suite par les actions de l'utilisateur et par les scripts.

La valeur initiale d'une commande ne change pas. Lorsqu'un formulaire est réinitialisé, la valeur courante de chaque commande redevient sa valeur initiale. Si la commande est dépourvue de valeur initiale, l'effet de la réinitialisation du formulaire sur cette commande est indéfini.

Lors de la soumission du formulaire pour son traitement, le nom et la valeur courante de certaines commandes sont accouplés. Ces couples sont soumis avec le formulaire.

HTML définit plusieurs grands types de commandes :

- **Boutons d'action.** Comme l'indique leur nom, les boutons d'action ont pour but de déclencher une action. Celle-ci peut être définie, comme pour un bouton de soumission (qui soumet le

formulaire) ou de réinitialisation (qui redonne leur valeur initiale à toutes les commandes), ou définie par l'utilisateur. Un formulaire peut contenir plusieurs boutons de soumission. Un bouton d'action défini par l'utilisateur ne possède pas de comportement par défaut. Chaque bouton poussoir peut avoir des scripts côté client qui sont associés aux attributs d'événement de l'élément. Quand un événement se produit (par exemple, l'utilisateur presse le bouton, le relâche, etc.), le script associé est déclenché. Le langage du script du bouton doit être défini à l'aide d'une déclaration de script par défaut (avec l'élément `META`). Un bouton est créé à l'aide d'un élément `BUTTON` ou `INPUT`.

- **Cases à cocher**. Les cases à cocher sont des interrupteurs marche/arrêt qui peuvent être actionnés par l'utilisateur. L'interrupteur est sur « marche » lorsque l'attribut `checked` de l'élément de commande est spécifié. Lors de la soumission du formulaire, seules les commandes de cases à cocher activées peuvent devenir des commandes réussies. Des cases à cocher peuvent être regroupées au sein d'un formulaire sous un même nom de commande. Elles permettent ainsi aux utilisateurs de sélectionner plusieurs valeurs pour la même propriété. Vous créez une case à cocher à l'aide de l'élément `INPUT`.
- **Boutons d'option**. Parfois également appelés boutons radio, les boutons d'option sont analogues aux cases à cocher. Ils en diffèrent toutefois en ce que, lorsqu'ils sont regroupés sous un même nom de commande, ils sont mutuellement exclusifs : lorsqu'un bouton est mis sur « marche », tous les autres boutons du groupe sont automatiquement mis sur « arrêt ». Vous créez un bouton radio à l'aide de l'élément `INPUT`. En raison des différences d'interprétation entre les agents utilisateurs, veillez à ce que l'un des boutons d'option d'un groupe soit mis initialement sur « marche ».
- **Menus**. Les menus proposent des options aux utilisateurs, parmi lesquelles il faut faire un choix. Vous créez un menu à l'aide de l'élément `SELECT` combiné avec des éléments `OPTGROUP` et `OPTION`.
- **Saisie de texte**. Deux types de commande permettent aux utilisateurs la saisie d'un texte. L'élément `INPUT` crée une commande pour une saisie sur une seule ligne, l'élément `TEXTAREA`, pour une saisie sur plusieurs lignes. Dans les deux cas, le texte saisi devient la valeur courante de la commande.

- **Sélection d'un fichier**. Ce type de commande permet à l'utilisateur de sélectionner un fichier de sorte que son contenu puisse être soumis avec le formulaire. Une commande de sélection de fichier est créée à l'aide de l'élément `INPUT`.
- **Commandes cachées**. Il est possible de créer des commandes qui ne sont pas restituées mais dont les valeurs sont soumises avec le formulaire. Cela sert généralement à enregistrer les informations entre les échanges client/serveur, qui seraient autrement perdues du fait de la nature « sans état » du protocole HTTP (voir le document [RFC2616]). Une commande cachée est créée à l'aide de l'élément `INPUT`.
- **Commandes d'objets**. Il est possible d'insérer des objets génériques dans les formulaires, de façon que les valeurs qui leur sont associées soient soumises en même temps que les autres commandes. L'élément `OBJECT` permet de créer une commande d'objet.

Les éléments utilisés pour créer les commandes apparaissent généralement dans un élément `FORM`. Ils peuvent toutefois apparaître en dehors de la déclaration de l'élément `FORM` lorsqu'ils servent à créer une interface utilisateur.

Élément INPUT

Vous avez dû remarquer que l'élément `INPUT` est de loin l'élément le plus employé dans un formulaire. Il permet en effet de créer des boutons d'action, des cases à cocher, des boutons d'option, la saisie d'un texte d'une ligne, la sélection d'un fichier et de créer une commande cachée.

Cet élément, dépourvu de balise de fermeture, possède la syntaxe générale suivante :

```
<INPUT type=valeur name=valeur value=valeur size=valeur>
```

L'attribut `type` est l'attribut le plus important : il spécifie le type de commande à créer. Cet attribut peut posséder de nombreuses valeurs légales, qui conditionnent la présence ou l'absence d'autres attributs.

La valeur par défaut de cet attribut est `text`. Cela crée une commande de saisie de texte sur une seule ligne. L'attribut complémentaire `size` indique à l'agent utilisateur la largeur initiale de la commande, exprimée

en nombre entier de caractères. L'attribut complémentaire `maxlength` spécifie le nombre maximum de caractères que l'utilisateur peut saisir. Par exemple,

```
Nom de l'utilisateur : <INPUT type=text name="nomutilisateur"
value="" size="10" maxlength="25">
```

crée un champ de 10 caractères de largeur, qui peut accepter jusqu'à 25 caractères en tout.



Figure 9.2 :
Exemple d'élément INPUT de type text

Lorsqu'il sera soumis, le nom du champ sera *nomutilisateur* et la valeur (`value`) sera ce qui a été saisi dans le champ.

La valeur `password` crée une commande de saisie de texte sur une seule ligne, de type mot de passe : le texte saisi est restitué de manière à dissimuler les caractères (par exemple, une succession d'astérisques). La valeur courante est le texte saisi par l'utilisateur, non le texte restitué par l'agent utilisateur. Vous employez en ce cas les attributs complémentaires `size` et `maxlength` comme avec la valeur `text`.

```
Mot de passe : <INPUT type="password" name="motdepasse"
size="10" maxlenght="15">
```

Cet exemple crée un champ de 10 caractères de largeur, qui peut accepter jusqu'à un maximum de 15 caractères en tout. Lors de la saisie, les caractères sont remplacés à l'écran par des astérisques.



Figure 9.3 :
Exemple d'élément INPUT de type password

Comme pour la valeur `text`, lors de la soumission, le nom du champ sera *motdepasse* et la valeur (`value`) ce qui a été saisi dans le champ.



Mot de passe

La protection offerte par ce dispositif reste toute relative. Bien qu'il soit masqué par l'agent utilisateur aux yeux d'un éventuel observateur, le mot de passe est transmis au serveur en texte clair et peut être lu par quiconque possède un accès au réseau.

La valeur `checkbox` crée une case à cocher, tandis que la valeur `radio` crée un bouton d'option. L'attribut complémentaire `value`, qui spécifie la valeur initiale de la commande, est alors obligatoire. L'attribut booléen `checked` spécifie que le bouton (ou la case) est sur « marche ». En voici un exemple :

```
<P>Sexe :<BR>
<INPUT type="radio" name="sexe" value="homme" checked> Homme
<INPUT type="radio" name="sexe" value="femme"> Femme
</P>
<P>Mat&eacute;riel poss&eacute;d&eacute; :<BR>
<INPUT type="checkbox" name="matos" value="LCD" checked>
  Ecran plat
<INPUT type="checkbox" name="matos" value="DVD">
  Graveur de DVD
<INPUT type="checkbox" name="matos" value="CD">
  Graveur de CD
<INPUT type="checkbox" name="matos" value="multi">
  Imprimante multifonction
</P>
```

Figure 9.4 : *Exemple d'éléments INPUT de type checkbox et radio*

Remarquez dans cette figure que le bouton **Homme** est coché, tout comme les cases *Écran plat* et *Graveur de DVD*.

Si chaque bouton d'option nécessite un élément séparé, vous les rassemblez dans un même groupe à l'aide d'un même `name`, ici « sexe ». Des boutons d'option appartenant au même groupe sont mutuellement exclusifs : un clic sur l'un d'entre eux désactive les autres. Vous pouvez posséder plusieurs groupes de boutons d'option, chaque groupe de boutons possédant un `name` unique. Chaque bouton possède toutefois une `value` différente, ici « homme » et « femme ». Homme et femme étant des valeurs mutuellement exclusives, ils constituent de bons candidats pour des boutons d'option.

En revanche, les cases à cocher laissent l'utilisateur libre de sélectionner plus d'une option dans un groupe. Les champs de cases à cocher d'un même groupe possèdent le même nom, des valeurs différentes, mais ne s'excluent pas mutuellement. Dans cet exemple, l'utilisateur peut posséder zéro ou plusieurs des matériels cités : des cases à cocher constituent donc un excellent choix.

Les valeurs `submit` et `reset` créent respectivement un bouton de soumission et un bouton de réinitialisation. Voici un exemple de mise en œuvre :

```
<INPUT type="submit" value="Envoyer !">
<INPUT type="reset" value="Annuler">
```



Figure 9.5 : *Exemple d'éléments INPUT de type submit et reset*

Ces boutons possèdent des significations préétablies : vous n'avez pas à leur lier de script. Le bouton de soumission effectue l'action définie dans l'élément `FORM`, tandis que le bouton de réinitialisation redonne à tous les éléments du formulaire leur valeur initiale. Exceptionnellement, l'attribut `name` est normalement superflu pour ces deux boutons : la valeur qui apparaît sur le bouton lui-même n'est pas envoyée avec le reste des données du formulaire. Elle ne possède pas de nom et est par conséquent sans intérêt.

Si vous spécifiez toutefois un nom pour votre bouton **Envoyer**, le nom et la valeur (`name` et `value`) de ce bouton seront envoyés. Cette caractéristique peut se révéler très utile en combinaison avec l'emploi de langages de script lors du traitement de votre formulaire.

La valeur `image` crée un bouton de soumission graphique. La valeur de l'attribut complémentaire `src` spécifie alors l'URI de l'image qui va décorer le bouton. Pour des questions d'accessibilité, mieux vaut toujours fournir un texte de remplacement pour l'image, à l'aide de l'attribut `alt`. L'attribut complémentaire obligatoire `src` spécifie la localisation de l'image à utiliser pour décorer le bouton de soumission graphique.

Lorsque vous cliquez sur l'image à l'aide d'un dispositif de pointage, le formulaire est soumis et les coordonnées du clic sont transmises au serveur. La coordonnée « x » se mesure en pixels à partir de la gauche de l'image et la coordonnée « y » en pixels à partir du haut de l'image. Les données soumises comprennent les valeurs `nom`.x=valeur–de–x et `nom`.y=`valeur–de–y`, dans lesquelles `nom` est la valeur de l'attribut `name`, et `valeur-de-x` et `valeur-de-y` respectivement les valeurs des coordonnées « x » et « y ».

Remarquez que, si le serveur doit entreprendre des actions différentes selon l'endroit cliqué, l'utilisateur d'un navigateur non graphique sera désavantagé. Mieux vaut donc recourir aux approches alternatives suivantes :

- Employer plusieurs boutons de soumission (chacun avec sa propre image) au lieu d'un seul bouton de soumission graphique. Les feuilles de style permettent de contrôler le positionnement de ces boutons.
- Employer une image cliquable côté client en combinaison avec des scripts.

Voici un exemple :

```
<INPUT type="image" src="images/boutonenvoyer.gif"
  alt="Envoyer !">
```

Il est créé un bouton de type image, qui emploie le fichier image *boutonenvoyer.gif* (situé comme il se doit dans le dossier *images* du site) et dont le texte alternatif est « Envoyer ! ».



Figure 9.6 :
Exemple d'élément INPUT de type image

La valeur `button` crée un bouton d'action. L'intitulé du bouton est la valeur de l'attribut `value`.

```
<INPUT type="button" name="fond"
  value="Changer la couleur de fond">
```

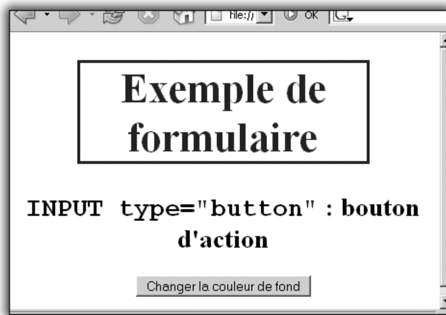


Figure 9.7 :
Exemple d'élément INPUT de type button

La valeur `hidden` crée une commande cachée.

```
<INPUT type="hidden" name="sourceform" value="feedback.html">
```

Nous ne montrons pas de copie d'écran, puisque par définition cette commande n'y apparaît pas !

Cela permet par exemple de savoir quel formulaire l'utilisateur a rempli, ou de soumettre l'heure, la date, les données de l'utilisateur, ou des données similaires en plus du reste des données du formulaire.

Enfin, la valeur `file` crée une commande de sélection de fichiers. Le nom du fichier initial est en principe la valeur de l'attribut `value`.

```
<B>Fichier &agrave; envoyer :</B>
<INPUT type=”file” name=”envoyer” size=10>
```

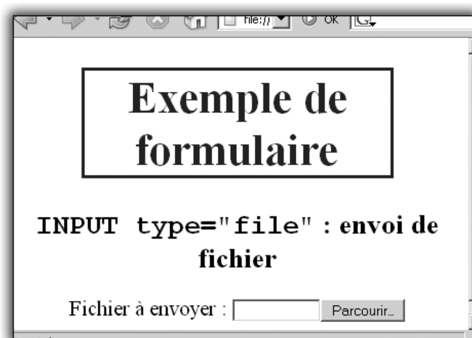


Figure 9.8 : *Exemple d'élément INPUT de type file*

Remarquez dans la figure précédente qu'un bouton **Parcourir** est automatiquement ajouté à côté du champ de saisie du nom de fichier : vous pouvez cliquer dessus pour parcourir votre système de fichiers et sélectionner le fichier concerné.

En spécifiant dans l'élément `FORM` l'attribut `enctype` et en lui attribuant la valeur `multipart/form-data`, chaque contenu de fichier sera conditionné pour soumission dans une section distincte d'un document en plusieurs parties.

Les autres attributs de l'élément `INPUT` sont présentés dans le tableau suivant.

Tableau 9.1 : *Autres attributs de l'élément INPUT*

Attribut	**Valeur**	**But**
`Checked`	*booléen*	Lorsque l'attribut `type` possède la valeur `radio` ou `checkbox`, spécifie que le bouton est sur « marche ». Ignoré pour les autres types de commande.
`Maxlength`	*entier*	Spécifie, lorsque l'attribut `type` possède la valeur `text` ou `password`, le nombre maximum de caractères que l'utilisateur peut saisir. Ce nombre peut excéder la valeur spécifiée pour l'attribut `size`, auquel cas l'agent utilisateur devrait proposer un mécanisme de défilement. La valeur par défaut de cet attribut est un nombre illimité.
`Name`	*nom*	Nom de la commande
`Notab`		Retire l'élément de l'ordre de tabulation

Tableau 9.1 : Autres attributs de l'élément INPUT		
Attribut	**Valeur**	**But**
`Size`	*entier*	Indique à l'agent utilisateur la largeur initiale de la commande. La largeur est donnée en pixels, sauf lorsque l'attribut `type` possède la valeur `text` ou `password`. Il s'agit alors d'un nombre entier de caractères.
`Src`	*uri*	Lorsque l'attribut `type` possède la valeur `image`, spécifie la localisation de l'image à utiliser pour décorer le bouton de soumission graphique.
`Tabindex`	*entier*	Définit la place dans l'ordre de tabulation
`Value`	*valeur*	Valeur initiale de la commande. Facultatif, sauf lorsque l'attribut `type` possède la valeur `radio` ou `checkbox`.

Nous reviendrons dans la suite de ce chapitre sur les attributs `tabindex` et `notab`.



Élément ISINDEX

L'élément `ISINDEX` n'est pas un élément de formulaire, puisqu'il n'appartient pas à un élément `FORM`. Son emploi est fortement déconseillé. Cet élément crée une commande de saisie d'un texte sur une seule ligne, qui admet un nombre quelconque de caractères. Les agents utilisateurs peuvent utiliser la valeur de l'attribut `prompt` de cet élément comme titre pour l'invite. Mieux vaut employer `INPUT` pour créer des commandes de saisie de texte.

Ainsi, la déclaration `ISINDEX` suivante :

```
<ISINDEX prompt="Saisissez le mot &agrave; rechercher : ">
```

pourrait se réécrire comme suit avec l'élément `INPUT` :

```
<FORM action="..." method="post">
<P>Saisissez le mot &agrave; rechercher : <INPUT type="text">
</P>
</FORM>
```

Élément BUTTON

Les boutons créés par l'élément `BUTTON` fonctionnent exactement comme ceux créés avec l'élément `INPUT`, mais offrent des possibilités de restitution plus variées, puisque l'élément `BUTTON` peut posséder un

contenu. Par exemple, un élément BUTTON qui contient une image fonctionne de la même façon et peut avoir le même aspect qu’un élément INPUT dont l’attribut type possède la valeur image, mais admet en outre un contenu.

Cet élément doit impérativement posséder une balise ouvrante et une balise de fermeture.

Les agents utilisateurs visuels peuvent restituer les boutons BUTTON en relief et avec un mouvement de haut en bas quand on les clique, les boutons INPUT étant restitués comme des images plates (reportez-vous à la figure de la page 378).

Ses principaux attributs sont les suivants :

- *name* spécifie le nom de la commande.
- *value* définit la valeur initiale du bouton.
- *type* déclare le type du bouton. Les valeurs possibles sont la valeur par défaut submit (bouton de soumission), reset (bouton de réinitialisation) et button (bouton d’action).

L’exemple suivant reprend et prolonge un exemple précédent en créant des boutons de soumission et de réinitialisation avec l’élément BUTTON au lieu de INPUT. Les boutons contiennent des images par l’intermédiaire d’éléments IMG.

```
<FORM action="http://unsite.fr/prog/ajoutermembre"
  method="post">
    <P>
    Pr&eacute;nom : <INPUT type="text" name="prenom"><BR>
    Nom : <INPUT type="text" name="nom"><BR>
    E-mail: <INPUT type="text" name="email"><BR>
    <INPUT type="radio" name="genre" value="homme"> Homme<BR>
    <INPUT type="radio" name="genre" value="femme"> Femme<BR>
    <BUTTON name="submit" value="envoyer" type="submit">
      <IMG src="images/c_bon.gif" alt="C’est tout bon !">
    </BUTTON>
    <BUTTON name="reset" type="reset">
      <IMG src="images/c_pas_bon.gif" alt="Surtout pas !">
    </BUTTON>
    </P>
</FORM>
```

Le résultat est présenté dans la figure suivante. Pensez à toujours fournir un texte de remplacement pour l’élément IMG.

Figure 9.9 :
Exemple d'élément BUTTON



IMG **et** BUTTON

Il est illégal d'associer une image cliquable à un élément IMG apparaissant en contenu d'un élément BUTTON. L'exemple suivant est interdit :

```
<BUTTON>
<IMG src="foo.gif" usemap="...">
</BUTTON>
```

Éléments SELECT, OPTGROUP et OPTION

Élément SELECT

Vous créez un menu à l'aide d'un élément SELECT. Cet élément possède obligatoirement une balise d'ouverture et une balise de fermeture. Chaque option du menu est représentée par un élément OPTION. Un élément SELECT doit contenir au moins un élément OPTION. Voici un exemple.

```
<P>Choisissez votre navigateur :
<SELECT NAME="navigateur">
    <OPTION value="nn">Netscape Navigator</OPTION>
    <OPTION selected value="ff">Firefox</OPTION>
    <OPTION value="msie">Microsoft Internet Explorer 5 ou
       ant&eacute;rieur</OPTION>
    <OPTION value="msie6">Microsoft Internet Explorer 6.x
    </OPTION>
```

```
    <OPTION value="autre">Texte ou autre</OPTION>
</SELECT>
```

Cet exemple crée ici un menu déroulant proposant les choix qui figurent dans les différents éléments OPTION. Ce choix de mode de présentation dépend toutefois de l’agent utilisateur.

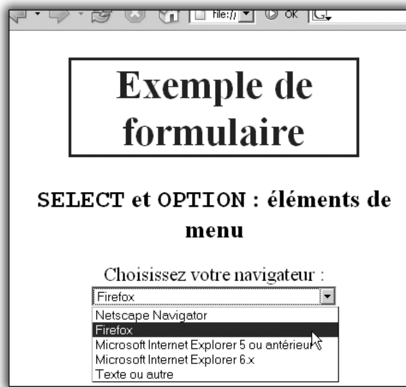


Figure 9.10 :
Exemple de menu (élément SELECT)

Les principaux attributs de l’élément SELECT sont les suivants :

- L’attribut name spécifie le nom de la commande.
- L’attribut size spécifie le nombre de lignes de la zone de liste déroulante, lorsque l’élément SELECT se présente comme tel. Les agents utilisateurs ne sont pas tenus de présenter l’élément SELECT sous forme d’une zone de liste : ils peuvent faire appel à un autre mécanisme, comme un menu déroulant.
- L’attribut booléen facultatif multiple permet une sélection multiple. En son absence, l’élément SELECT n’autorise qu’une sélection unique.

Lorsque l’attribut multiple est spécifié, les utilisateurs peuvent choisir plusieurs options en maintenant enfoncée la touche Ctrl pendant qu’ils cliquent sur chacun de leurs choix.

Élément OPTION

Chaque choix d’un menu est représenté par un élément OPTION. Seule la balise d’ouverture est obligatoire, mais, comme nous l’avons déjà signalé, mieux vaut toujours fermer un élément HTML.

Les principaux attributs de l'élément OPTION sont les suivants :

- L'attribut label permet de spécifier un intitulé pour l'option plus court que le contenu de l'élément OPTION. Les agents utilisateurs doivent employer la valeur de cet attribut, lorsqu'il est présent, plutôt que le contenu de l'élément OPTION comme intitulé de l'option.
- L'attribut facultatif value spécifie la valeur initiale de la commande. S'il est absent, la valeur initiale est le contenu de l'élément OPTION.
- L'attribut booléen facultatif selected spécifie que l'option est présélectionnée. Dans l'exemple précédent, nous avions décidé que le choix *Firefox* était présélectionné :

  ```
  <OPTION selected value="ff">Firefox</OPTION>
  ```

Il est possible de présélectionner zéro ou plusieurs options. Les agents utilisateurs devraient déterminer les options présélectionnées comme suit :

- Si aucun élément OPTION ne possède d'attribut selected, le comportement de l'agent utilisateur vis-à-vis du choix de l'option sélectionnée initiale est indéfini.
- Si un élément OPTION possède un attribut selected, alors celui-ci devrait être présélectionné.
- Si l'élément SELECT possède un attribut multiple spécifié et plusieurs éléments OPTION avec l'attribut selected spécifié, alors ceux-ci devraient tous être présélectionnés.

Si plusieurs éléments OPTION possèdent un attribut selected alors que l'attribut multiple n'est pas spécifié sur l'élément SELECT, c'est une erreur. Les agents utilisateurs peuvent varier dans la prise en charge de cette erreur, mais ils ne devraient pas présélectionner plus d'une option.

Élément OPTGROUP

Vous pouvez regrouper logiquement des options à l'aide de l'élément OPTGROUP. Cela est particulièrement utile quand l'utilisateur doit effectuer un choix à partir d'une longue liste d'options : les groupes d'options apparentées sont plus faciles à comprendre et à se remémorer qu'une seule longue liste d'options. Dans HTML 4, les groupes imbriqués sont interdits : tous les éléments OPTGROUP doivent être spécifiés directement dans un élément SELECT.

Les balises d'ouverture et de fermeture de l'élément `OPTGROUP` sont obligatoires. Son principal attribut est `label`, qui spécifie l'intitulé du groupe d'options.



Regroupement

Des futures versions de HTML pourraient (et devraient) étendre le mécanisme de regroupement, de façon à autoriser l'imbrication de groupes d'options.

Voici un exemple d'emploi de l'élément `OPTGROUP` :

```
<SELECT name="navigateur">
<OPTGROUP label="Nescape Navigator">
   <OPTION label="4-" value="NN4-">
     Ant&eacute;rieur &agrave; la version 4.0</OPTION>
   <OPTION label="4-6" value="NN4_6">Version 4 &agrave; 6.x
   </OPTION>
   <OPTION label="6+" value="NN6+">Post&eacute;rieur &agrave;
   la version 6.x</OPTION>
</OPTGROUP>
<OPTGROUP label="Firefox" selected>
   <OPTION label="1.0" value="FF1">Firefox 1.0</OPTION>
   <OPTION label="1.1" value="FF11" selected>Firefox 1.2
   </OPTION>
   <OPTION label="1.2" value="FF12">Firefox 1.2</OPTION>
</OPTGROUP>
<OPTGROUP label="Internet Explorer">
   <OPTION label="4-" value="IE4-">
    Ant&eacute;rieur &agrave; la version 4.0</OPTION>
  <OPTION label="4-6" value="IE4_6">Version 4 &agrave; 6
   </OPTION>
   <OPTION label="6+" value="IE6+">Post&eacute;rieur
    &agrave; la version 6.x</OPTION>
</OPTGROUP>
<OPTION selected label="autre" value="autre">Autre</OPTION>
</SELECT>
```

Les agents utilisateurs visuels peuvent autoriser les utilisateurs à effectuer une sélection à partir des groupes d'options au moyen d'un menu hiérarchique ou d'un autre mécanisme reflétant la structure des options : c'est le cas de Firefox, comme le montre la figure suivante (voir Figure 9.11).

Remarquez que chaque sous-menu affiche l'intitulé d'un élément `OPTGROUP` ou d'un élément `OPTION`.

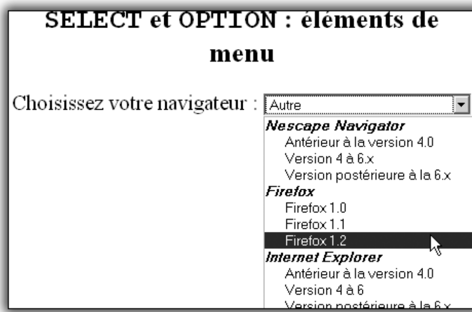


Figure 9.11 : *Exemple de menu, avec recours à l'élément OPTGROUP.*

Élément TEXTAREA

L'élément `TEXTAREA` permet de créer une zone de texte de saisie multiligne. Vos utilisateurs pourront la remplir en soumettant leurs commentaires, suggestions ou messages de forum. `TEXTAREA` permet à l'utilisateur de saisir plusieurs lignes de texte. Celles-ci sont contenues dans `value`. Les agents utilisateurs doivent employer comme valeur initiale le contenu de cet élément, intégralement restitué. Les balises d'ouverture et de fermeture de cet élément sont obligatoires.

Les attributs de l'élément `TEXTAREA` sont les suivants :

- L'attribut `name` spécifie le nom de la commande.
- L'attribut `rows` spécifie le nombre de lignes de texte visibles. Les utilisateurs doivent pouvoir saisir plus de lignes que ce nombre : les agents utilisateurs devraient donc fournir un moyen de faire défiler le contenu de la commande quand celui-ci s'étend au-delà de la zone visible.
- L'attribut `cols` spécifie la largeur visible en fonction de la chasse moyenne des caractères. Les utilisateurs doivent pouvoir saisir des lignes plus longues que cette largeur. Les agents utilisateurs peuvent couper les textes de ligne visibles afin de garder les longues lignes visibles sans devoir les faire défiler.

Voici un exemple :

```
<FORM action="http://unsite.fr/prog/lecture-texte"
  method="post">
   <P>
   <TEXTAREA name="le_texte" rows="10" cols="80">
   Premi&egrave;re ligne de texte initial.
```

```
    Seconde ligne de texte initial.
    </TEXTAREA>
    <INPUT type="submit" value="Envoyer"><INPUT type="reset">
    </P>
</FORM>
```

Cet exemple crée une commande `TEXTAREA` de dix lignes sur quatre-vingts colonnes et qui contient initialement deux lignes de texte. La commande `TEXTAREA` est suivie par deux boutons de soumission et de réinitialisation.

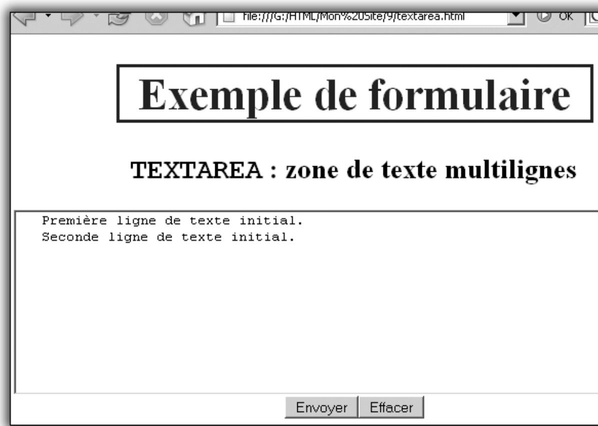


Figure 9.12 : *Exemple d’élément TEXTAREA*

La spécification de l’attribut `readonly` permet à l’auteur d’afficher un texte non modifiable dans la commande `TEXTAREA`. Cela diffère de l’emploi d’un texte balisé standard dans un document, puisque la valeur de l’élément `TEXTAREA` est soumise avec le formulaire.

Élément LABEL

Quelques commandes de formulaire possèdent des libellés ou *labels* automatiquement associés (comme les boutons d’action), mais la plupart en sont dépourvues (les champs de texte, les cases à cocher, les boutons d’option ainsi que les menus).

Lorsqu’une commande possède un libellé implicite, les agents utilisateurs doivent employer la valeur de l’attribut `value` comme chaîne de caractères du libellé.

L'élément LABEL permet de spécifier un libellé pour les commandes dépourvues de libellé implicite. Chaque élément LABEL est associé à exactement une commande de formulaire. Ses balises d'ouverture et de fermeture sont obligatoires.

L'attribut for associe explicitement un label à une autre commande : la valeur de l'attribut for doit être la même que celle de l'attribut id d'une certaine commande du même document. Quand il est absent, le libellé qui est défini est associé au contenu de l'élément.

Dans l'exemple suivant, un tableau permet d'aligner deux commandes de saisie de texte ainsi que les libellés qui leur sont associés. Chaque libellé est associé explicitement à une commande de saisie de texte :

```
<FORM action="..." method="post">
<TABLE>
  <TR>
    <TD><LABEL for="libelle_prenom">Pr&eacute;nom</LABEL>
    <TD><INPUT type="text" name="prenom" id="libelle_prenom">
  <TR>
    <TD><LABEL for="libelle_nom">Nom</LABEL>
    <TD><INPUT type="text" name="nom" id="libelle_nom">
</TABLE>
</FORM>
```

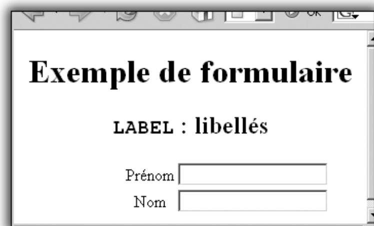


Figure 9.13 :
Exemple d'éléments LABEL



L'attribut for

Il est possible d'associer plusieurs éléments LABEL à la même commande en créant plusieurs références *via* l'attribut for.

L'exemple complique le précédent, en présentant plus d'éléments LABEL :

```
<FORM action="http://unsite.fr/prog/ajoutermembre"
  method="post">
     <TABLE style="margin-left: 30%">
```

```
      <TR>
         <TH><LABEL for="libelle_prenom">Pr&eacute;nom :
             </LABEL></TH>
         <TD><INPUT type="text" id="libelle_prenom"></TD>
     </TR>
     <TR>
        <TH><LABEL for="libelle_nom">Nom : </LABEL></TH>
        <TD><INPUT type="text" id="libelle_nom"></TD>
     </TR>
     <TR>
       <TH><LABEL for="libelle_email">Adrelle : </LABEL></TH>
       <TD><INPUT type="text" id="libelle_email"></TD>
     </TR>
     <TR>
       <TD colspan="2">
         <INPUT type="radio" name="sexe" value="homme"> Homme
         <BR>
         <INPUT type="radio" name="sexe" value="femme"> Femme
         </TD>
     </TR>
   </TABLE>
   <P>
      <INPUT type="submit" value="Envoyer"> <INPUT
        type="reset">
   </P>
</FORM>
```

Figure 9.14 :
Exemple d'éléments LABEL

Pour associer implicitement un libellé à une autre commande, l'élément de commande doit se trouver à l'intérieur de l'élément LABEL. Cet élément LABEL ne peut alors contenir qu'un seul élément de commande. Le label en question peut se placer avant ou après la commande associée.

L'exemple suivant associe implicitement deux libellés à deux commandes de saisie de texte :

```
<FORM action="..." method="post">
<P>
<LABEL>
   Pr&eacute;nom
   <INPUT type="text" name="prenom">
</LABEL>
<LABEL>
   <INPUT type="text" name="nom">
   Nom
</LABEL>
</P>
</FORM>
```

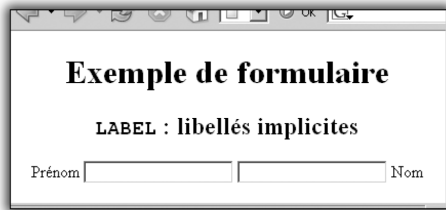


Figure 9.15 :
Exemple d'éléments LABEL (libellés implicites)

Remarquez qu'il est impossible d'employer cette technique lorsque la disposition est assurée par un tableau : le libellé se trouve dans une cellule et la commande associée dans une autre.

Lorsqu'un élément `LABEL` reçoit le focus, celui-ci est communiqué à la commande associée. Le focus sera étudié dans la suite de ce chapitre.

Les agents utilisateurs peuvent restituer les libellés de nombreuses façons (par exemple : visuellement, lus par des synthétiseurs de parole, etc.).

Ajout d'une structure à un formulaire : éléments FIELDSET et LEGEND

Comme nous l'avons déjà largement souligné, une des vocations profondes de HTML est de se préoccuper de la structure des documents. Cette notion de structure semble quelque peu mise à mal par les différents éléments de formulaires…

Heureusement, l'élément `FIELDSET` remédie à cette situation. Il permet de regrouper par thème les commandes et les libellés apparentés. Le regroupement des commandes facilite leur compréhension par les utilisateurs, tout en améliorant la navigation par tabulation pour les agents utilisateurs visuels et la navigation vocale pour les agents

utilisateurs vocaux. La bonne utilisation de cet élément rend plus accessibles les documents.

L'élément LEGEND permet aux auteurs d'assigner une légende à un élément FIELDSET. Il renforce l'accessibilité lorsque l'élément FIELDSET est restitué de manière non visuelle. Ses balises d'ouverture et de fermeture sont obligatoires.

Le principal attribut de LEGEND est align, désormais déconseillé au profit des feuilles de style. Il positionne la légende par rapport au jeu de champs. Ses valeurs possibles sont la valeur par défaut top (au-dessus du jeu de champs), bottom (en dessous), left (à gauche) et right (à droite).

L'exemple suivant propose un formulaire de type « recherche de logiciel ». Il se divise en trois parties : les informations personnelles, le système d'exploitation et la catégorie recherchée. Chaque partie contient les commandes pour la saisie des informations concernées.

```
<FIELDSET>
<LEGEND>Syst&egrave;me d'exploitation</LEGEND>
  <INPUT name="systeme_expl" type="radio"
    value="Windows" checked> Windows
  <INPUT name="systeme_expl" type="radio"
    value="Mac OX" > Mac OX
  <INPUT name="systeme_expl" type="radio"
    value="Linux"> Linux
  <INPUT name="systeme_expl" type="radio"
    value="autre"> Autre
</FIELDSET>
 <FIELDSET>
  <LEGEND>Cat&eacute;gorie recerch&eacute;e</LEGEND>
  Quel type de produit recherchez-vous ?
  <INPUT name="categorie" type="checkbox"
    value="utilitaire">Utilitaire syst&egrave;me
  <INPUT name="categorie" type="checkbox"
    value="sonvideo">Son et vid&eacute;o
  <INPUT name="categorie" type="checkbox"
    value="bureautique">Bureautique
<BR>
  Si la cat&eacute;gorie recherchée ne figure pas
  dans la liste pr&eacute;c&eacute;dente, saisissez
  ci-dessous des mots-cl&eacute;s s&eacute;par&eacute;s
  par des virgules.
<BR>
  <TEXTAREA name="recherche" rows="3" cols="80">
  </TEXTAREA>
 </FIELDSET>
</FORM>
```

La figure suivante montre l'aspect de ce formulaire sous Firefox.

Figure 9.16 : *Exemple d'éléments FIELSET et LEGEND*

Remarquez qu'il serait possible d'améliorer à l'aide de la feuille de style la présentation visuelle du formulaire, en alignant les éléments à l'intérieur de chaque élément `FIELDSET` et en ajoutant de la couleur et des indications de police. Vous pourriez également ajouter des scripts, par exemple pour n'afficher la zone de texte *Recherche* que si l'utilisateur indique ne pas trouver de catégorie, etc.

9.2. Formulaire et focus

Dans un document HTML, un élément doit recevoir le focus (aussi nommé « attention ») grâce à une action de l'utilisateur pour devenir actif et remplir sa fonction. Par exemple, un utilisateur doit activer (en général cliquer) le lien spécifié par l'élément `A` pour suivre le lien en question. De la même manière, les utilisateurs doivent donner le focus à l'élément `TEXTAREA` pour y saisir un texte.

Il existe plusieurs façons de donner le focus à un élément :

- En désignant l'élément avec un dispositif de pointage.

- En naviguant d'un élément à l'autre au clavier. Il peut avoir été défini un ordre de tabulation spécifiant l'ordre dans lequel les éléments reçoivent le focus quand l'utilisateur navigue à l'aide du clavier dans le document. Une fois sélectionné, l'élément peut être activé par une certaine combinaison de touches.
- En sélectionnant l'élément au moyen d'une touche d'accès rapide (appelée aussi parfois « raccourci clavier » ou « clé d'accès »).

Navigation par tabulation

Dans un document HTML, plusieurs éléments sont soumis à un ordre de tabulation, explicite ou implicite. Il s'agit essentiellement des éléments qui reconnaissent l'attribut `tabindex`, à savoir les éléments `A`, `AREA`, `BUTTON`, `INPUT`, `OBJECT`, `SELECT` et `TEXTAREA`.

L'ordre de tabulation définit l'ordre dans lequel les éléments recevront le focus lorsque l'utilisateur naviguera à l'aide du clavier. L'ordre de tabulation peut comprendre des éléments imbriqués dans d'autres éléments.

Cet ordre de tabulation est défini de façon explicite à l'aide de l'attribut `tabindex`, pour les éléments qui le reconnaissent. Sa syntaxe est la suivante :

```
tabindex = nombre
```

`nombre` doit être compris entre 0 et 32 767. Les agents utilisateurs doivent ignorer les éventuels zéros de tête.

Les agents utilisateurs parcourent les éléments qui peuvent recevoir le focus selon les règles suivantes :

1. D'abord les éléments qui reconnaissent l'attribut `tabindex` et lui assignent une valeur positive. La navigation part de l'élément dont l'attribut `tabindex` possède la plus petite valeur pour aller vers l'élément qui possède la valeur la plus élevée. Les valeurs ne se suivent pas forcément, ni ne doivent commencer à une valeur particulière. Les éléments dont les valeurs de l'attribut `tabindex` sont identiques sont parcourus dans l'ordre de leur apparition dans le flux de caractères du code.
2. Ceux des éléments qui ne reconnaissent pas l'attribut `tabindex`, ou bien le reconnaissent et lui assignent une valeur `0`, sont

parcourus ensuite. Ces éléments sont parcourus dans l'ordre de leur apparition dans le flux de caractères du code.

Cela signifie que si aucun élément ne possède d'attribut `tabindex`, l'ordre de tabulation sera celui de l'apparition des éléments dans le code.

Les éléments inactifs ne participent pas dans l'ordre de tabulation.

Examinez le listing suivant :

Listing 9-1 : tabindex.html

```
<!DOCTYPE HTML PUBLIC
   "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
   <HEAD>
      <TITLE>Exemple de formulaire</TITLE>
      <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
        charset=iso-8859-1">
      <META name="version" content="9.1">
      <META http-equiv="Content-Style-Type" content="text/css">
      <LINK rel="STYLESHEET" type="text/css" href="style.css">
   </HEAD>
   <BODY>
      <H1>Exemple de formulaire</H1>
      <H2>Navigation &agrave; l'aide de la touche
       <CODE>TAB</CODE></H2>
      <P>...un texte...</P>
      <P>Consultez le
         <A tabindex="10" href="http://www.w3.org/">
           site Web du W3C.</A>
         <BR>
...suite du texte...
         <BR>
         <BUTTON type="button" name="action" tabindex="1"
           onclick="action()">
         Effectuer l'action pr&eacute;vue.
         </BUTTON>
        <BR>
        ...suite du texte...
        <BR>
      <FORM action="..." method="post">
         <P>
            <INPUT tabindex="1" type="text" name="champs1">
            <INPUT tabindex="2" type="text" name="champs2">
            <BR>
            <INPUT tabindex="3" type="submit" name="submit">
        </P>
   </FORM>
```

```
  </BODY>
</HTML>
```

Dans cet exemple, l'ordre de tabulation est le suivant : l'élément `BUTTON`, les éléments `INPUT` dans l'ordre (remarquez que celui nommé *champs1* partage la même valeur d'attribut `tabindex` que le bouton, mais *champs1* apparaît plus tard dans le flux de caractères) et enfin le lien créé par l'élément `A`.

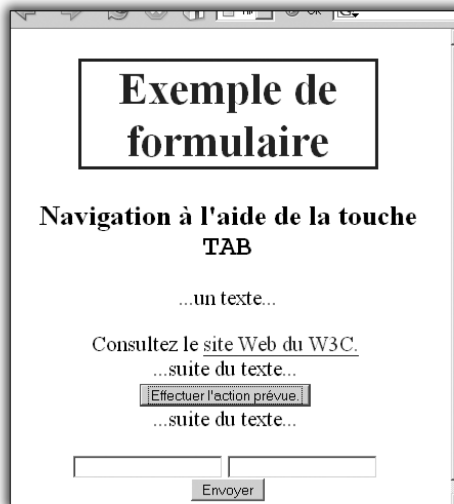


Figure 9.17 : *Navigation à l'aide de l'attribut tabindex*



Touches configurables

La touche ou la combinaison de touches qui produit la navigation par tabulation dépend de la configuration de l'agent utilisateur. C'est généralement la touche [Tab] pour la navigation et la touche [Entrée] pour l'activation de l'élément sélectionné, mais cela n'est en rien une obligation impérative.

Les agents utilisateurs peuvent également définir des combinaisons de clés pour parcourir l'ordre de tabulation à l'envers. Quand la fin ou le début de l'ordre de tabulation est atteint, l'agent utilisateur peut revenir en arrière au début ou à la fin.

Touches d'accès rapide

Vous êtes probablement familier avec les touches d'accès rapide de votre système d'exploitation ou de vos logiciels favoris. Par exemple, vous savez certainement qu'avec Windows la combinaison ctrl+c copie un objet, tandis que la combinaison ctrl+v colle cet objet à l'emplacement actif. HTML permet de définir de telles touches d'accès rapide pour un élément, à l'aide de l'attribut `accesskey`. Sa syntaxe est la suivante :

```
accesskey = caractère
```

`caractère` est un caractère seul qui provient du jeu de caractères du document.



Clé d'accès

Pensez à tenir compte de la méthode de saisie de l'utilisateur supposé lors de la spécification d'une clé d'accès.

Un appui sur la touche d'accès rapide affectée à un élément lui procure le focus. Ce qui se passe lorsque l'élément reçoit le focus dépend de l'élément. Par exemple, si l'utilisateur active un lien défini par un élément `A`, l'agent utilisateur suit en général le lien. Lorsque l'utilisateur active un bouton radio, l'agent utilisateur change la valeur du bouton radio. Quand l'utilisateur active un champ de texte, la saisie devient possible, etc.

Vous pouvez définir un attribut `accesskey` pour les éléments suivants : `A`, `AREA`, `BUTTON`, `INPUT`, `LABEL`, `LEGEND` et `TEXTAREA`.

L'exemple qui suit affecte la touche d'accès rapide N au libellé associé à une commande `INPUT`. Un appui sur la touche procure le focus au libellé, qui le transmet à la commande associée. L'utilisateur peut alors saisir un texte dans la zone `INPUT`.

```
<FORM action="..." method="post">
<P>
<LABEL for="libelle_utilisateur" accesskey="N">
Nom d'utilisateur
</LABEL>
<INPUT type="text" name="nom_utilisateur"
  id="libelle_utilisateur">
</P>
</FORM>
```

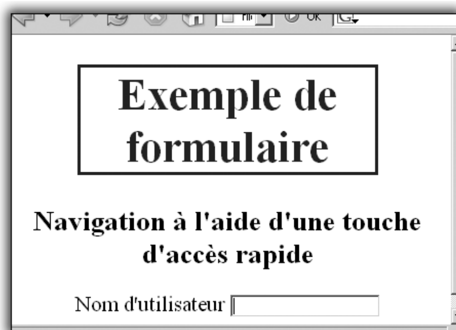


Figure 9.18 :
Touches d'accès rapide

Compliquons cet exemple en ajoutant un lien hypertexte lorsque l'utilisateur ne possède pas de compte. Une touche d'accès rapide est affectée à ce lien défini par un élément A. Un appui sur cette touche mène l'utilisateur vers ce qui serait un document offrant un formulaire d'ouverture de compte.

```
<FORM action="..." method="post">
... idem ci-dessus ...
</FORM>
<P>
<LABEL>
<A rel="contents" accesskey="C" href="inscription.html">
    Cr&eacute;er un compte
</A>
</LABEL>
</P>
```

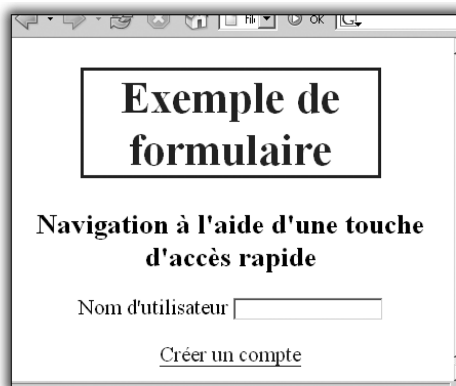


Figure 9.19 :
Touche d'accès rapide



Touches d'accès rapide

La procédure exacte d'invocation des touches d'accès rapide dépend du système d'exploitation sous-jacent. Avec MS Windows, c'est en général une combinaison alt+*touche d'accès rapide*. Avec un système Apple, ce serait cmd+*touche d'accès rapide*.

La restitution des clés d'accès est fonction de l'agent utilisateur. Mieux vaut inclure la clé d'accès dans le texte du libellé ou partout où la clé d'accès doit s'appliquer. Les agents utilisateurs devraient restituer la valeur d'une clé d'accès de façon à mettre son rôle en évidence et à la distinguer des autres caractères (par exemple, en la soulignant).

9.3. Commandes inactives et en lecture seule

Parfois, une saisie de l'utilisateur peut être indésirable ou superflue. Il faut donc pouvoir rendre une commande inactive ou la restituer en lecture seule. Vous pourriez par exemple souhaiter que le bouton de soumission d'un formulaire reste inactif tant que l'utilisateur n'a pas entré certaines données obligatoires. De la même manière, l'auteur peut vouloir inclure un bout de texte en lecture seule, qui doit être soumis comme valeur en même temps que le formulaire. Les sections suivantes décrivent les commandes inactives et celles en lecture seule.

Commandes inactives

Vous déclarez une commande comme inactive à l'aide de l'attribut booléen `disabled`. La commande est désactivée et refuse toute forme d'entrée de l'utilisateur. Cet attribut peut être employé avec les éléments `BUTTON`, `INPUT`, `OPTGROUP`, `OPTION`, `SELECT` et `TEXTAREA`.

Lorsqu'il est présent dans un élément, l'attribut `disabled` produit l'effet suivant :

- Les commandes inactives ne reçoivent pas le focus.
- Les commandes inactives sont ignorées au cours d'une navigation par tabulation.
- Les commandes inactives ne peuvent pas réussir.

Cet attribut est hérité, les déclarations locales surclassant toutefois la valeur héritée.

Dans l'exemple suivant, l'élément INPUT est inactif. Il ne peut pas recevoir d'entrée de l'utilisateur et sa valeur ne peut pas être soumise avec le formulaire.

```
<INPUT type="submit" disabled name="Soumettre"">
```

La façon dont sont restitués les éléments inactifs dépend de l'agent utilisateur. Par exemple, certains agents utilisateurs, comme Firefox, restituent en « grisé » les articles de menu, les labels de bouton, etc., inactifs.



Figure 9.20 :
Commande désactivée

Seul un script peut modifier dynamiquement la valeur de l'attribut disabled.

Commandes en lecture seule

L'attribut booléen readonly , lorsqu'il est présent dans une commande de formulaire, interdit les modifications de la commande. Vous pouvez l'employer avec les éléments INPUT et TEXTAREA.

Quand il est présent, l'attribut readonly produit les effets suivants sur l'élément :

- Les éléments en lecture seule reçoivent l'attention mais les utilisateurs ne peuvent pas les modifier.
- Les éléments en lecture seule sont inclus dans la navigation par tabulation.
- Les éléments en lecture seule peuvent réussir.

La façon dont sont restitués les éléments en lecture seule dépend de l'agent utilisateur.

Voici un exemple :

```
Syst&egrave;me d'exploitation : <INPUT name="systeme_expl"
  type="text" value="Windows" readonly >
```

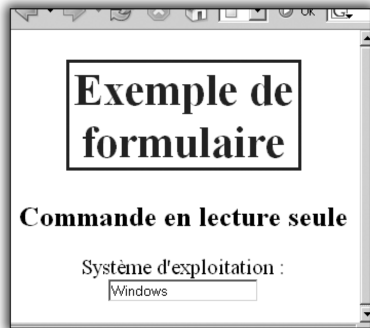


Figure 9.21 :
Commande en lecture seule

Il est impossible de modifier le contenu de la zone de texte présentée dans la figure précédente.

Seul un script peut modifier dynamiquement la valeur de l'attribut `readonly`.

9.4. Soumission du formulaire

Dans tout ce qui précède, nous avons examiné avec soin les éléments grâce auxquels il était possible de faire saisir des informations par l'utilisateur. Le formulaire rempli, l'utilisateur clique sur le bouton **Soumettre**. Les sections suivantes expliquent la manière dont les agents utilisateurs soumettent les données de formulaire aux agents qui traitent les formulaires.

Méthodes de soumission du formulaire

L'attribut `method` de l'élément `FORM` spécifie la méthode HTTP employée pour envoyer le formulaire à l'agent de traitement. Cet attribut admet deux valeurs :

- `get` : l'ensemble des données du formulaire est rajouté à l'URI spécifié par l'attribut `action` (avec comme séparateur un caractère point d'interrogation `?`) et ce nouvel URI est envoyé à l'agent de traitement.
- `post` : l'ensemble des données de formulaire est inclus dans le corps du formulaire et envoyé à l'agent de traitement.

La méthode `get` est à préférer lorsque le formulaire ne produit aucun effet secondaire (il est alors dit « idempotent »). La plupart des recherches dans une base de données sont dépourvues d'effets secondaires visibles et font des applications idéales pour la méthode `get`. Les moteurs de recherche emploient `get` : une demande de recherche n'ajoute rien à un fichier ou ne modifie en rien les données du serveur.

Si le service associé au traitement d'un formulaire entraîne des effets secondaires (par exemple, si le formulaire modifie une base de données ou l'abonnement à un service), mieux vaut employer la méthode `post`. Elle est ainsi employée pour les pages d'inscription, les salles de discussion et les forums de discussion en ligne.



Le codage des caractères avec `Get` et `Post`
La méthode `get` restreint les valeurs du jeu des données du formulaire aux caractères ASCII. En revanche, la méthode `post`, lorsqu'elle est spécifiée avec l'attribut `enctype="multipart/form-data"`, recouvre la totalité du jeu de caractères ISO 10646.

Commandes réussies

Une commande est dite « réussie » lorsque son nom est apparié à sa valeur courante et que ce couple appartient au jeu des données du formulaire qui est soumis. Une commande réussie peut être soumise. Une commande réussie doit être définie dans un élément `FORM` et posséder un nom de commande.

Remarquez toutefois que :

- Les commandes inactives ne peuvent réussir.
- Si le formulaire contient plusieurs boutons de soumission, seul le bouton de soumission actif réussira.

- Toutes les cases à cocher activées (sur « marche ») peuvent réussir.
- Dans le cas de boutons d'option qui partagent le même nom (la même valeur pour l'attribut `name`), seul le bouton d'option activé peut réussir.
- Pour les menus, le nom de commande est donné par l'élément `SELECT` et les valeurs sont fournies par les éléments `OPTION`. Seules les options sélectionnées peuvent réussir. Si aucune option n'est sélectionnée, la commande échoue : ni le nom ni aucune valeur ne sont soumis au serveur avec le formulaire.
- La valeur courante d'une sélection de fichiers est une liste d'un ou plusieurs noms de fichiers. Lors de la soumission du formulaire, le contenu de chaque fichier est soumis avec le restant des données du formulaire. Les contenus des fichiers sont conditionnés en fonction du type de contenu du formulaire.
- La valeur courante d'une commande d'objet est déterminée par l'implémentation de l'objet.

Si une commande est dépourvue de valeur courante au moment de la soumission du formulaire, les agents utilisateurs ne sont pas obligés de la traiter comme une commande réussie.

En outre, les agents utilisateurs ne doivent pas considérer les commandes suivantes comme étant réussies :

- Les boutons de réinitialisation.
- Les éléments `OBJECT` dont l'attribut `declare` n'est pas spécifié.

Les commandes cachées et les commandes qui ne sont pas restituées en raison de l'effet d'une feuille de style peuvent quand même réussir. Par exemple :

```
<FORM action="..." method="post">
<P>
<INPUT type="password" style="display:none"
          name="mot_de_passe_invisible"
          value="mon_mot_de_passe">
</FORM>
```

Cela entraîne malgré tout l'accouplement de la valeur au nom `mot_de_passe_invisible` et leur soumission avec le formulaire.

Traitement des données du formulaire

Lorsque l'utilisateur soumet le formulaire (par exemple, en activant un bouton de soumission), l'agent utilisateur le traite de la manière suivante.

1. Première étape : identification des commandes réussies.
2. Deuxième étape : construction du jeu des données du formulaire (le jeu des données du formulaire est la séquence des couples « nom de commande/valeur courante » construite à partir des commandes réussies).
3. Troisième étape : codage du jeu des données du formulaire.

   Le jeu des données du formulaire est alors codé en fonction du type de contenu spécifié par l'attribut `enctype` de l'élément `FORM`.
4. Quatrième étape : soumission du jeu des données du formulaire codé.
5. Enfin, les données codées sont envoyées à l'agent de traitement désigné par l'attribut `action`, en utilisant le protocole spécifié par l'attribut `method`.

Cette spécification ne définit pas toutes les méthodes de soumission valides ni les types de contenu qui peuvent être employés avec les formulaires. Les agents utilisateurs HTML 4 doivent cependant obéir aux conventions établies dans les cas suivants :

- Si la valeur de l'attribut `method` est `get` et la valeur de l'attribut `action` est un URI HTTP, alors l'agent utilisateur prend la valeur de l'attribut `action`, lui rajoute un caractère `?` et enfin le jeu des données du formulaire, codé en utilisant le type de contenu `application/x-www-form-urlencoded`. L'agent utilisateur traverse alors le lien vers cet URI. Dans ce scénario, les données du formulaire se limitent aux codes ASCII.
- Si la valeur de l'attribut `method` est `post` et la valeur de l'attribut `action` est un URI HTTP, alors l'agent utilisateur conduit une transaction HTTP `post` en utilisant la valeur de l'attribut `action` et un message créé en fonction du type de contenu spécifié par l'attribut `enctype`.

Pour toute autre valeur de l'attribut `action` ou `method`, le comportement n'est pas spécifié.

Les agents utilisateurs devraient restituer les réponses des transactions HTTP `get` et `post`.

Types de contenu du formulaire

L'attribut `enctype` de l'élément `FORM` spécifie le type de contenu utilisé pour coder le jeu des données du formulaire en vue de sa soumission au serveur. Les agents utilisateurs doivent reconnaître les types de contenu listés ci-dessous. Le comportement pour d'autres types de contenu n'est pas spécifié.

Le type de contenu par défaut est `application/x-www-form-urlencoded`. Les formulaires soumis avec ce type de contenu doivent être codés comme suit :

1. Les noms de commandes et les valeurs sont échappées. Les caractères « espace » sont remplacés par des caractères plus (`+`) puis les caractères réservés sont échappés comme décrit dans le document [RFC1738], section 2.2. Les caractères non alphanumériques sont remplacés par une séquence de la forme `%HH` : un caractère pourcentage et deux chiffres hexadécimaux qui représentent le code ASCII du caractère en question. Les sauts de ligne sont représentés par des couples de caractères `CR LF` (`%0D%0A`).
2. Les couples nom/valeur des commandes sont listés selon leur ordre d'apparition dans le document. Le nom est séparé de la valeur par un caractère égal `=`, et les couples nom/valeur sont séparés les uns des autres par des esperluettes (`&`).

Le type de contenu `application/x-www-form-urlencoded` est inefficace pour l'envoi de grandes quantités de données binaires ou de texte contenant des caractères non ASCII.

L'autre valeur possible est `multipart/form-data`. C'est le type de contenu à employer pour la soumission de formulaires contenant des fichiers, des données non ASCII et des données binaires.

9.5. Remarques à propos des formulaires

Avant de créer un formulaire sur votre site, la première chose qui vous vient à l'esprit est de dégager les questions qui vont être posées, et celles qui seront obligatoires. Il est en effet possible d'empêcher la soumission d'un formulaire si un des champs obligatoires n'est pas rempli. Ceci est souvent utilisé sur le Web pour exiger un nom ou une adresse.

Figure 9.22 :
Champs de formulaire obligatoires, dans le formulaire d'enregistrement du site de MicroApplication.

Cela n'est toutefois qu'un aspect du problème. Il est plus important encore de savoir de quelles informations vous avez réellement besoin et comment expliquer ce fait à l'utilisateur. Certains sites exigent beaucoup plus d'informations que nécessaire. Est-il réellement indispensable de demander son numéro de téléphone à un utilisateur pour obtenir simplement des commentaires sur votre site ? Toute information possède une valeur, et les utilisateurs deviennent de plus en plus méfiants sur les données qu'ils fournissent gratuitement. Ils peuvent en outre ne pas savoir qui vous êtes exactement, ou pourquoi vous voulez cette information.

Vous ne devez demander et à plus forte raison n'exiger que les informations strictement nécessaires à l'objectif de votre formulaire. Si la raison pour laquelle vous voulez ces informations n'est pas instantanément évidente, alors expliquez vos motivations. Si vous conservez des données utilisateur dans un fichier, vous devez déclarer celui-ci à la CNIL (*Commission Nationale Informatique et Liberté*).

Un formulaire HTML ne présente que peu ou pas d'intérêt en l'absence d'un dispositif capable de traiter les informations recueillies. Ce traitement est généralement effectué à l'aide de scripts, qui feront l'objet du prochain chapitre.

9.6. Résumé

- Un formulaire constitue un moyen d'apporter un peu d'interactivité à une page HTML.
- Vous créez un formulaire dans une page HTML à l'aide de l'élément `FORM`. Cet élément agit comme conteneur général du formulaire, notamment pour les commandes.
- Les utilisateurs interagissent avec les formulaires au moyen de commandes nommées. Le « nom » d'une commande est spécifié par son attribut `name`. Chaque commande possède à la fois une *valeur initiale* et une *valeur courante*. La *valeur initiale* d'une commande peut généralement être spécifiée à l'aide de l'attribut `value` de l'élément de commande.
- HTML définit plusieurs grands types de commandes : boutons d'action, cases à cocher, boutons d'option, menus, saisie de texte et sélection d'un fichier.
- L'élément `INPUT` est le plus employé dans un formulaire. Il permet de créer des boutons d'action, des cases à cocher, des boutons d'option, la saisie d'un texte d'une ligne et la sélection de fichiers. Son attribut `type` spécifie le type de la commande.
- La valeur `text` est la valeur par défaut de cet attribut. Elle crée une commande de saisie de texte sur une seule ligne. La valeur `password` crée une commande de saisie de texte sur une seule ligne de type « mot de passe » : le texte saisi est restitué de manière à dissimuler les caractères. La valeur `checkbox` crée une case à cocher, tandis que la valeur `radio` crée un bouton d'option. Les valeurs `submit` et `reset` créent respectivement un bouton de soumission et un bouton de réinitialisation. La valeur `image` crée un bouton de soumission graphique. La valeur `button` crée un bouton d'action. La valeur `hidden` crée une commande cachée. Enfin, la valeur `file` crée une commande de sélection de fichiers.
- Les boutons créés par l'élément `BUTTON` fonctionnent exactement comme ceux créés avec l'élément `INPUT`, mais offrent des possibilités de restitution plus variées, puisque l'élément `BUTTON` peut posséder un contenu.
- Vous créez un menu à l'aide d'un élément `SELECT`. Chaque option du menu est représentée par un élément `OPTION`. Un élément `SELECT` doit contenir au moins un élément `OPTION`.

- Vous pouvez regrouper logiquement des options à l'aide de l'élément `OPTGROUP`.
- L'élément `TEXTAREA` permet de créer une zone de texte de saisie multiligne.
- L'élément `LABEL` permet de spécifier un libellé pour les commandes dépourvues de libellé implicite.
- L'élément `FIELDSET` permet de regrouper par thème les commandes et les libellés apparentés.
- L'élément `LEGEND` permet aux auteurs d'assigner une légende à un élément `FIELDSET`.
- Vous pouvez donner le focus à un élément de plusieurs façons : à l'aide d'un dispositif de pointage, en naviguant d'un élément à l'autre au clavier selon l'ordre de tabulation ou à l'aide de touches d'accès rapide.
- L'ordre de tabulation est défini de façon explicite à l'aide de l'attribut `tabindex`. Une touche d'accès rapide est définie pour un élément à l'aide de l'attribut `accesskey`.
- Vous déclarez une commande comme inactive à l'aide de l'attribut booléen `disabled`.
- L'attribut booléen `readonly`, lorsqu'il est présent dans une commande de formulaire, interdit les modifications de cette commande.
- L'attribut `method` de l'élément `FORM` spécifie la méthode HTTP employée pour envoyer le formulaire à l'agent de traitement. Cet attribut admet deux valeurs : `get` et `post`.
- Une commande est dite « réussie » lorsque son nom est apparié à sa valeur courante et que ce couple appartient au jeu des données du formulaire qui est soumis.
- Les données sont envoyées à l'agent de traitement désigné par l'attribut `action`, en utilisant le protocole spécifié par l'attribut `method`.
- L'attribut `enctype` de l'élément `FORM` spécifie le type de contenu utilisé pour coder le jeu des données du formulaire en vue de sa soumission au serveur. Il existe deux types de contenu reconnus : par défaut, `application/x-www-form-urlencoded` et `multipart/form-data`, à employer pour la soumission de formulaires contenant des fichiers, des données non ASCII et des données binaires.

Chapitre 10

Scripts

Introduction aux scripts 381
Élément SCRIPT 383
Événements intrinsèques 390
Travail avec les scripts 396
Applets 424
Résumé 432

Vous savez désormais créer une page Web d'une relative complexité sans trop de difficultés. Vous savez y incorporer des graphismes et y adjoindre des liens. Vous allez dans ce chapitre apprendre à doter vos pages Web de plus de dynamisme grâce à l'emploi de scripts.

Un script est un programme écrit dans un langage de programmation particulier. Son exécution est déclenchée par un événement qui se produit sur la machine cliente, soit automatiquement (comme au chargement d'un document HTML) soit suite à une action de l'utilisateur.

Il existe deux types de scripts : les scripts côté serveur et les scripts côté client.

- **Les scripts côté serveur** sont situés et s'exécutent comme leur nom l'indique sur le serveur, et peuvent envoyer un résultat vers la machine cliente. Ils étaient anciennement de type CGI *(Common Gateway Interface), une* technique qui permettait de lancer un programme sur une machine en utilisant les entrées de celle d'un client. D'autres scripts côté serveur comprennent des images cliquables côté serveur et des extensions ou API brevetées, comme ASP (*Active Server Pages*), ColdFusion, FrontPage Extensions ou le très populaire PHP.
- **Les scripts côté client** sont des programmes qui peuvent accompagner le document HTML ou bien y être directement incorporés. Ils s'exécutent sur la machine cliente (préservant ainsi les ressources du serveur) également suite à un événement. Nous reviendrons par la suite sur les événements HTML. HTML gère les scripts d'une façon indépendante du langage de script. Ces scripts sont généralement rédigés en JavaScript ou en VBScript, et nécessitent de l'utilisateur un navigateur capable et autorisés à lire et à exécuter le langage de script que vous utilisez, quel qu'il soit. Les scripts côté client sont donc très dépendants du navigateur, mais ils sont aussi bien plus rapides que ceux côté serveur.

Ce chapitre s'intéresse essentiellement aux scripts côté client.

Éléments et attributs étudiés dans ce chapitre :

```
SCRIPT, type, language, src, charset, defer
NOSCRIPT
APPLET, code, width, height
OBJECT, codetype, codebase, classid, data, type, archive,
  declare, standby
PARAM, name, value, valuetype
```

10.1. Introduction aux scripts

Les scripts permettent d'augmenter la réactivité et le dynamisme de documents HTML. Par exemple :

- Modifier dynamiquement le contenu d'un document lors du chargement de celui-ci.
- Accompagner un formulaire et valider les données avant la soumission.
- Actions particulières suite à un événement comme le chargement, le déchargement, la prise de focus d'un élément, le mouvement de la souris, etc.
- Action liée aux commandes d'un formulaire (par exemple, les boutons) pour simuler une interface utilisateur graphique.

Cette liste permet de séparer les scripts en deux catégories principales :

- **Scripts à action unique** : ils sont exécutés une seule fois, lors du chargement du document par l'agent utilisateur. Ils apparaissent dans le document à l'intérieur d'un élément `SCRIPT`. Pour tenir compte des agents utilisateurs qui ne peuvent pas gérer les scripts, mieux vaut inclure un contenu de remplacement *via* l'élément `NOSCRIPT`.

Ces scripts sont capables de modifier dynamiquement le contenu du document. Leurs capacités dépendent du langage de script concerné. La modification dynamique d'un document peut être modélisée comme suit :

- Tous les éléments `SCRIPT` sont évalués dans l'ordre au fur et à mesure du chargement du document.
- Toutes les structures de script à l'intérieur d'un élément `SCRIPT` donné, qui génèrent des valeurs SGML de type `CDATA`, sont évaluées. Leur texte généré combiné est inséré dans le document à la place de l'élément `SCRIPT`.
- Les données générées de type `CDATA` sont réévaluées.

Par exemple, imaginons le fragment de document suivant (vous l'étudierez plus en détail par la suite) :

```
<script language="JavaScript">
<!--
Navigateur = navigator.appName;
```

```
if (Navigateur == 'Netscape')
{
document.write("<TD><A href='region_nets.html' ");
}
if (Navigateur == 'Microsoft Internet Explorer')
{
document.write("<TD><A href='region_IE.html' ");
}
document.writeln("target=princ>Ma r&eacute;gion</A></TD>"
//-->
</script>
```

Considérons que vous l'examinez avec un navigateur de type Netscape, configuré pour accepter les scripts.

1 Le navigateur examine le contenu de l'élément `SCRIPT`.

2 Comme il s'agit d'un navigateur de type Netscape, c'est le premier cas de l'instruction conditionnelle qui est évalué. Le moteur de script place donc dans le document la chaîne de caractères `<TD><A href='region_nets.html'`, sort de l'instruction conditionnelle, puis insère la chaîne `target =princ>Ma région</A></TD>` et sort de l'élément `SCRIPT`. Le navigateur interprète ensuite la chaîne totale, soit :

```
<TD><A href='region_IE.html' target=princ>Ma r&eacute;gion
</A></TD>
```

S'il s'était agi d'un navigateur Internet Explorer, le résultat aurait été :

```
<TD><A href='region_nets.html' target=princ>Ma r&eacute;gion
</A></TD>
```

Les documents HTML sont contraints de se conformer au DTD HTML avant comme après le traitement de chaque élément `SCRIPT`.

- **Scripts répétitifs** : ils sont exécutés chaque fois qu'un événement particulier se produit. Ils peuvent être affectés à un certain nombre d'éléments à l'aide des attributs d'*événement intrinsèque*, étudiés plus loin.

HTML ne dépend pas d'un langage de script particulier. Les auteurs des documents indiquent explicitement aux agents utilisateurs le langage de chaque script.

Les deux grands langages de script côté client, JavaScript (développé par Netscape) et VBScript (Microsoft), ont mené une bataille féroce afin

d'attirer l'attention des développeurs. Microsoft a même développé sa propre version de JavaScript, nommée JScript. Cela s'est traduit à l'époque par une fragmentation croissante des sites Web : une fonction n'était reconnue que par un type de navigateur et partageait le public en deux parties, ceux qui possédaient un navigateur compatible et tous les autres.

Tout ce qui impose aux concepteurs Web de faire un choix quant à la plate-forme de destination est une négation de la technologie et de l'esprit qui régit Internet et le World Wide Web. L'idée originelle était et reste de partager idées et informations, indépendamment de l'ordinateur, du système d'exploitation ou de l'application.

La tendance est toutefois désormais au respect de ECMAScript, une tentative de normalisation du noyau du langage JavaScript et de ses dérivés : sa syntaxe, ses mots-clés et ses composants natifs. La troisième édition du standard ECMA-262 a été publiée en décembre 1999 (**www.ecma-international.org/publications/standards/Ecma-262.htm**).

Nous reviendrons plus en détail sur ces principaux langages dans la suite de ce chapitre. Pour le moment, intéressons-nous à la façon d'incorporer un script côté client à un document HTML.

10.2. Élément SCRIPT

Vous insérez un script dans un document à l'aide de l'élément `SCRIPT`, obligatoirement doté de balises d'ouverture et de fermeture. Cet élément peut apparaître un nombre quelconque de fois dans les éléments `HEAD` ou `BODY` d'un document HTML.

Par exemple, ajoutons à notre page d'accueil un petit script :

1. Ouvrez *accueil.html* dans le Bloc-Notes.
2. Modifiez le numéro de version :

```
<META name="version" content="3.10.1">
```

3. Insérez, dans l'élément `H2` du corps (l'élément `BODY`), l'élément `SCRIPT` suivant, en remplaçant ce qui se trouve entre la balise d'ouverture `H2` et l'élément `EM` :

```
<H2>
<SCRIPT type="text/javascript">
<!--
```

```
    document.write("Bienvenue sur ")
</SCRIPT>
-->
<EM>mon</EM> site.</H2>
```

4 Enregistrez votre fichier, sans modifier son nom, puis ouvrez *index.html* dans un navigateur. Guère passionnant : l'effet est strictement identique à la version précédente (regardez la Figure 10.1 et comparez-la à la Figure 8.10).

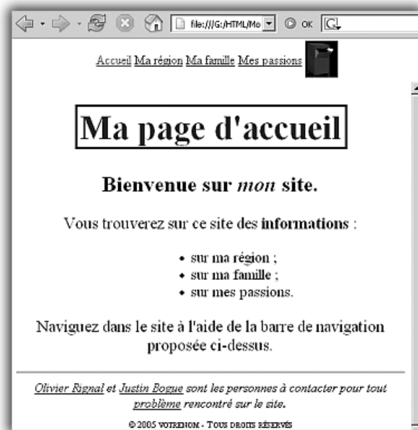


Figure 10.1 : *Votre premier script JavaScript*

Ce script JavaScript (remarquez la valeur de l'attribut `type`) est en effet d'une grande simplicité, puisqu'il ne contient qu'une instruction : `document.write` signifie « imprimer à l'écran l'information placée entre parenthèses ». Celle-ci étant identique au texte original, aucune différence n'est visible. Autrement dit, ce script est strictement identique à la version précédente du code, soit :

```
<H2>Bienvenue sur <EM>mon</EM> site.</H2>
```

Nous reviendrons par la suite sur la signification des lignes <!–– et ––>, mais souvenez-vous qu'il s'agit d'un commentaire HTML.



JavaScript désactivé

Aucune différence n'est visible si, et seulement si, votre navigateur est configuré de façon à autoriser les scripts. Si ce n'est pas le cas, vous obtenez quelque chose de similaire à ce qui est présenté figure suivante.



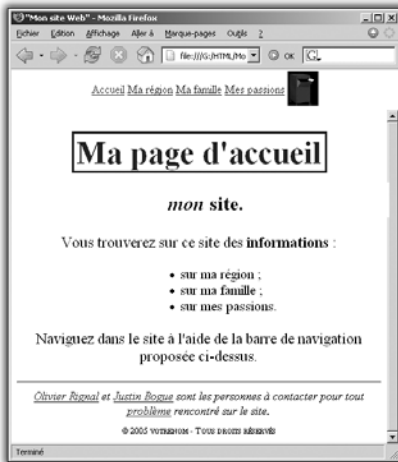


Figure 10.2 :
Page d'accueil sous Firefox, JavaScript étant désactivé.



Activer JavaScript

Vous activez JavaScript avec Firefox en choisissant **Outils** > **Options**. Sélectionnez **Général** dans la zone de gauche, puis cochez l'option *Autoriser JavaScript*.

Avec Internet Explorer, les réglages s'effectuent en choisissant **Outils** > **Options Internet**, puis en cliquant sur l'onglet **Sécurité**. Reportez-vous pour plus de détails à la documentation de votre navigateur. Internet Explorer 6 vous informe si une page contient un ou plusieurs script alors que le navigateur n'est pas configuré pour les accepter.

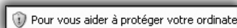


Figure 10.3 :
Message d'Internet Explorer, qui signale que les scripts sont désactivés.

Ce script ne présentant guère d'intérêt, essayons autre chose :

1. Revenez à *accueil.html*, ouvert dans le Bloc-Notes.
2. Enregistrez le fichier sous le nom *accueil_3_10_1.html*.
3. Modifiez le numéro de version :

```
<META name="version" content="3.10.2">
```

4. Redonnez à l'élément `H2` sa forme antérieure :

```
<H2>Bienvenue sur <EM>mon</EM> site.</H2>
```

5 Insérez après cet élément le code suivant :

```
<SCRIPT type="text/JavaScript">
<!--
document.write("<P>Vous utilisez un navigateur "
  + navigator.appName + "</P>");
 -->
</SCRIPT>
```

6 Enregistrez votre fichier sous le nom *accueil.html*, puis ouvrez *index.html* dans un navigateur. Le résultat est plus intéressant.



Figure 10.4 : *Affichage du type de navigateur à l'aide d'un script JavaScript*

Remarquez dans cette figure que Firefox est identifié comme navigateur Netscape. Il est possible d'obtenir plus de détails, en particulier sur la version employée, comme vous le verrez par la suite.

Définition du type de script : déclaration META et attribut type de l'élément SCRIPT

Nous ne le répéterons jamais assez : HTML ne dépend pas d'un langage de script particulier. Vous devez indiquer explicitement aux agents utilisateurs le langage de chaque script.

Vous pouvez spécifier un langage de script par défaut pour tous les scripts d'un document. Pour ce faire, placez la déclaration `META` suivante dans l'élément `HEAD` :

```
<META http-equiv="Content-Script-Type"
  content="un_certain_type">
```

La valeur `un_certain_type` représente le type de contenu qui nomme le langage de script, par exemple `text/tcl`, `text/JavaScript` ou `text/vbscript`.

Un document qui ne spécifie pas de langage de script par défaut et qui contient un ou plusieurs éléments spécifiant un ou plusieurs scripts d'événement intrinsèque est incorrect. Les agents utilisateurs peuvent toujours essayer d'interpréter les scripts spécifiés incorrectement, mais ne sont nullement tenus de le faire.

Vous pouvez également spécifier localement le langage d'un script en définissant l'attribut `type` de son élément `SCRIPT` dans le document. La valeur de l'attribut `type` spécifie le langage de script du contenu de l'élément. Il outrepasse le langage de script par défaut. Le langage de script est spécifié comme type de contenu (par exemple, `text/javascript`). Il faut fournir une valeur pour cet attribut, qui est dépourvu de valeur par défaut. Cet attribut remplace désormais l'attribut `language`, qui spécifiait le langage de script du contenu de cet élément à l'aide d'un identifiant du langage. Ces identifiants n'étant pas normalisés, l'attribut est déconseillé en faveur de l'attribut `type`.

Autres attributs de l'élément SCRIPT

Un script peut être défini en contenu de l'élément `SCRIPT` (comme dans les exemples précédents) ou dans un fichier externe. L'attribut `src` spécifie alors l'emplacement du script externe :

```
<SCRIPT type="text/javascript" src="quelquepart/script.js">
```

Si l'attribut `src` n'est pas spécifié, l'agent utilisateur doit interpréter le contenu de l'élément comme étant le script. Dans le cas contraire, l'agent utilisateur doit ignorer le contenu de l'élément et ramener le script désigné par l'URI. Remarquez que l'attribut optionnel `charset` se réfère à l'encodage de caractères du script désigné par l'attribut `src` et non au contenu de l'élément `SCRIPT`.

Quand il est présent, l'attribut booléen `defer` indique à l'agent utilisateur que le script ne va générer aucun contenu de document (par exemple, aucune instruction `document.write` en JavaScript). L'agent utilisateur peut donc poursuivre l'analyse et la restitution.

Cas des agents utilisateurs qui ne gèrent pas les scripts

Les scripts sont précieux, mais il serait dommage de fonder un site sur l'exploitation de ceux-ci sans tenir compte des agents utilisateurs qui ne reconnaissent pas les scripts. Deux précautions sont à prendre :

- Fournir un contenu de remplacement aux agents utilisateurs qui ne reconnaissent pas les scripts ou ne sont pas configurés pour les accepter.
- Dissimuler le contenu de ceux-ci pour les agents utilisateurs qui seraient susceptibles d'afficher leur contenu comme du texte.

Nous allons examiner ces deux précautions.

Élément NOSCRIPT

L'élément NOSCRIPT permet de fournir un contenu de remplacement lorsque, pour une raison quelconque, un script n'est pas exécuté. Le contenu de l'élément NOSCRIPT ne doit être restitué par un agent utilisateur reconnaissant les scripts que dans les cas suivants :

- L'agent utilisateur est configuré pour ne pas exécuter (évaluer) les scripts.
- L'agent utilisateur ne reconnaît pas le langage de script invoqué par un élément SCRIPT apparu plus tôt dans le document.

Les agents utilisateurs qui ne gèrent pas les scripts côté client doivent restituer le contenu de cet élément.

Dans l'exemple suivant, l'agent utilisateur qui interprète l'élément SCRIPT va inclure certaines données créées dynamiquement dans le document. S'il ne reconnaît pas les scripts, l'utilisateur peut toujours obtenir les données grâce à un lien.

```
<SCRIPT type="text/javascript">
 ...un script javascrit qui incorpore des donn&eacute;es...
</SCRIPT>
<NOSCRIPT>
   <P><A href="http://quelquepart.fr/données.html">
        Acc&egrave; aux donn&eacute;es.
      </A>
</NOSCRIPT>
```

Grâce à l'élément NOSCRIPT, vous pouvez fournir de précieuses indications à vos visiteurs.

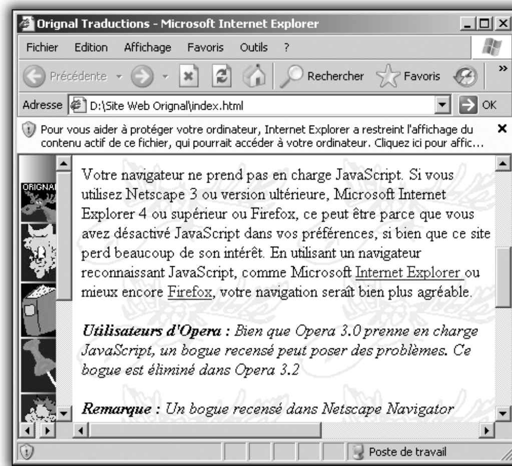


Figure 10.5 :
Exemple de contenu d'un élément NOSCRIPT

Dissimulation des données de script aux agents utilisateurs

La seconde précaution consiste à dissimuler le contenu des scripts. En effet, un agent utilisateur qui ne reconnaît pas l'élément SCRIPT va vraisemblablement restituer le contenu de cet élément comme un texte. Certains moteurs de script, dont ceux des langages JavaScript et VBScript, autorisent la délimitation des déclarations d'un script par un commentaire SGML <!-- -->. Les agents utilisateurs qui ne reconnaissent pas l'élément SCRIPT ignoreront le commentaire tandis que les moteurs de script intelligents comprendront que le script dans le commentaire doit être exécuté.



Solution alternative

Une autre solution à ce problème consiste à placer les scripts dans des documents externes et à les appeler avec l'attribut src.

Avec JavaScript, vous pouvez employer la chaîne <!-- au début de l'élément SCRIPT : le moteur ignore les caractères suivants jusqu'à la fin de la ligne. JavaScript interprète la chaîne // comme le début d'un

commentaire qui s'étend jusqu'à la fin de la ligne courante. Elle est nécessaire pour dissimuler la chaîne —> à l'analyseur JavaScript (même si celle-ci ne lui pose généralement pas de problème).

Voici un exemple :

```
<SCRIPT type="text/javascript">
<!—dissimulation du contenu du script aux anciens navigateurs
  function square(i) {
    document.write("L'appel a pass&eacute; ", i ,
                   " &agrave; la fonction.","<BR>")
    return i * i
  }
  document.write("La fonction a renvoy&é ",square(i),".")
// fin de la dissimulation du contenu -->
</SCRIPT>
```

Avec VBScript, c'est le caractère « guillemet simple » qui fait, du reste de la ligne courante, un commentaire. Par exemple :

```
<SCRIPT type="text/vbscript">
  <!--
    Sub foo()
     ...
    End Sub
  ' -->
 </SCRIPT>
```

Vous l'avez compris, le choix du caractère de commentaire dépend du langage de script employé. Avec Tcl, par exemple, ce serait le caractère `#`, etc.



Fermeture de commentaires

Certains navigateurs referment les commentaires au premier caractère > rencontré. Pour dissimuler le contenu du script à ces navigateurs, vous pouvez transposer les opérandes des opérateurs relationnels et de décalage (par exemple, en écrivant `y < x` plutôt que `x > y`) ou employer le mécanisme d'échappement du langage de script pour le caractère >.

10.3. Événements intrinsèques

Chaque langage de script obéit à ses propres règles pour appeler des objets HTML à partir d'un script. La spécification HTML ne définit aucun mécanisme standard d'appel des objets HTML. La syntaxe des données du script dépend du langage de script.

Les scripts doivent toutefois référer à un élément en fonction du nom qui lui est assigné. Les moteurs de script doivent observer la règle de préséance suivante dans l'identification d'un élément : l'attribut `name` est prioritaire sur l'attribut `id` si les deux sont présents. Sinon, l'un ou l'autre peuvent être employés indifféremment.

Il est possible d'associer une action à un certain nombre d'événements, qui se produisent lorsque l'utilisateur interagit avec l'agent utilisateur. Chacun des *événements intrinsèques* prend comme valeur un script :

```
nom_événement="script"
```



Evénements intrinsèques

Des modifications vont probablement être apportées aux événements intrinsèques, notamment en ce qui concerne la façon dont les scripts sont rattachés aux événements. Les recherches dans ce domaine sont menées par les membres du groupe de travail sur le Modèle Objet de Document (DOM) du W3C. Consultez le site Web du W3C (www.w3.org/) pour plus de renseignements.

Le script s'exécute chaque fois que l'événement se produit pour cet élément. La syntaxe des données de script dépend du langage de script.

Les événements reconnus par les différents éléments HTML sont présentés dans le tableau suivant.

Tableau 10.1 : *Événements intrinsèques HTML*

Événement	Survient lorsque
Événements concernant les éléments `BODY` et `FRAMESET`	
`onload`	L'agent utilisateur finit de charger une fenêtre ou bien tous les cadres dans un jeu d'encadrement `FRAMESET`
`onunload`	L'agent utilisateur retire le document d'une fenêtre ou d'un cadre
Événements de formulaire (élément `FORM`)	
`onsubmit`	Un formulaire est soumis
`onreset`	Un formulaire est réinitialisé
Événements concernant les commandes de formulaire	

***Tableau 10.1 :** Événements intrinsèques HTML*	
Événement	**Survient lorsque**
`onfocus`	Un élément reçoit le focus *via* le dispositif de pointage ou une navigation par tabulation. S'applique aux éléments `A`, `AREA`, `LABEL`, `INPUT`, `SELECT`, `TEXTAREA` et `BUTTON`.
`onblur`	Un élément perd le focus *via* le dispositif de pointage ou une navigation par tabulation. S'applique aux mêmes éléments que `onfocus`.
`onselect`	L'utilisateur sélectionne un certain texte dans un champ textuel. S'applique aux éléments `INPUT` et `TEXTAREA`.
`onchange`	Une commande perd le focus ou sa valeur a été modifiée depuis l'instant où elle a reçu le focus. S'applique aux éléments `INPUT`, `SELECT` et `TEXTAREA`.
Événements génériques. Tous les éléments, sauf `APPLET`, `BASE`, `BASEFONT`, `BDO`, `BR`, `FONT`, `FRAME`, `FRAMESET`, `HEAD`, `HTML`, `IFRAME`, `ISINDEX`, `META`, `PARAM`, `SCRIPT`, `STYLE` et `TITLE`.	
`onclick`	Le bouton du dispositif de pointage est cliqué au-dessus d'un élément
`ondblclick`	Le bouton du dispositif de pointage est double-cliqué au-dessus d'un élément
`onmousedown`	Le bouton du dispositif de pointage est appuyé au-dessus d'un élément
`onmouseup`	Le bouton du dispositif de pointage est relâché au-dessus d'un élément
`onmouseover`	Le dispositif de pointage est déplacé sur un élément
`onmousemove`	Le dispositif de pointage est déplacé alors qu'il est au-dessus d'un élément
`onmouseout`	Le dispositif de pointage est déplacé en dehors d'un élément
`onkeypress`	Une touche est pressée puis relâchée au-dessus d'un élément
`onkeydown`	Une touche est gardée appuyée au-dessus d'un élément
`onkeyup`	Une touche est relâchée au-dessus d'un élément

Comme vous pouvez le voir, la liste des événements qu'il est possible d'identifier et de relier à une action est impressionnante. Vous pouvez par exemple employer des éléments de commande de formulaire (`INPUT`, `SELECT`, `BUTTON`, `TEXTAREA` et `LABEL`), qui répondent tous à

certains événements intrinsèques, pour améliorer l'interface utilisateur graphique du document.

Voici quelques exemples d'emploi d'événements intrinsèques.

Nous avions vu, lors de l'examen des formulaires, que certains champs pouvaient être obligatoires, et qu'il était possible de tester la validité des données d'un formulaire avant sa soumission à l'aide de scripts.

Dans le premier exemple, la commande nommée `nomUtilisateur` est un champ obligatoire. Lorsque l'utilisateur quitte ce champ, celui-ci perd le focus (événement `onblur`), ce qui appelle une fonction JavaScript qui teste la validité de `nomUtilisateur`.

```
<INPUT name="nomUtilisateur"
       onblur="verifNomUtilisateur(this.value)">
```

Dans l'exemple suivant, le champ *nombre* doit normalement contenir une valeur comprise entre `1` et `10`. Le code JavaScript suivant vérifie ce fait après la saisie d'un nombre par l'utilisateur :

```
<INPUT name="nombre"
    onchange="if (!verifNombre(this.value, 1, 10))
        {this.focus();this.select();} else {remercier()}"
    value="0">
```



`Document.write`

Une déclaration `document.write` ou équivalente placée dans un gestionnaire d'événements intrinsèques crée et écrit vers un nouveau document au lieu de modifier le document courant.

Voici enfin un autre exemple d'emploi d'événements intrinsèques. Supposons que nous ayons modifié la barre de navigation en remplaçant le texte par une icône. Pour l'icône correspondant à *Accueil*, ce pourrait être une icône nommée *accueil.gif*.

Le fichier de la barre de navigation a été modifié en conséquence (ce fichier se trouve dans les fichiers exemples sous le nom *barrenav1.html* ; vous devez l'appeler à l'aide du fichier *index1.html*, également présent) :

```
<TD><A href="accueil.html" target=princ>
      <IMG src="images/accueil.gif" alt="Accueil" border="0">
      </A>
 </TD>
```

L'aspect de la barre de navigation, ainsi modifiée, serait quelque chose d'analogue à ce qui est présenté dans la figure suivante.



Figure 10.6 : *Aspect de la barre de navigation modifiée*

Imaginez que nous voulions que l'utilisateur visualise encore mieux sa navigation en modifiant l'aspect de l'icône. Pour faire simple, nous nous sommes bornés ici à inverser les couleurs de l'icône, à l'aide de la fonction **Négatif** de Paint Shop Pro, pour créer un fichier nommé *accueil2.gif*.



Figure 10.7 : *Aspect des icônes accueil.gif et accueil2.gif*

Nous modifions alors comme suit l'élément `A` qui concerne la page d'accueil, en ajoutant ce qui suit :

```
<TD><A href="accueil.html" target=princ
     onmouseover="t=acc.src;acc.src=acc.lowsrc"
     onmouseout="acc.src=t"
     >
<IMG name="acc" lowsrc="images/accueil2.gif"
     src="images/accueil.gif" alt="Accueil" border="0">
</A>
</TD>
```

Ce fichier se trouve dans les fichiers exemples, sous le nom *barrenav2.html*. En l'examinant dans votre navigateur (en appelant *index2.html*, également fourni), vous constatez que, lorsque votre dispositif de pointage (en principe votre souris) est placé au-dessus de l'icône *Accueil*, celle-ci se modifie immédiatement : cela ajoute de l'interactivité et du dynamisme à votre site, pour des efforts somme toute relativement faibles.



Figure 10.8 : Modification à la volée de l'icône selon que le dispositif de pointage est placé ou non dessus

Ce petit fragment de code mérite que vous l'examiniez d'un peu plus près, pour comprendre ce qui se passe. Il est un peu particulier en ce qu'il n'emploie pas un « vrai » script (vous remarquerez l'absence d'élément `SCRIPT`), mais a recours au Modèle Objet de Document (DOM) sous-jacent à tout document HTML ou XML bien formé. Regardez d'abord l'élément `IMG` : vous l'avez nommé (`name="acc"`), avez laissé l'attribut `src` défini comme précédemment, mais avez également défini un autre attribut, `lowsrc`, avec comme valeur la seconde icône.

Dans l'élément `A`, vous avez ajouté deux gestionnaires d'événements : un pour `onmouseover` (la souris est sur l'icône) et un pour `onmouseout` (la souris quitte l'icône).

Par défaut, la souris n'est pas sur l'icône et l'icône affichée est celle qui correspond à la valeur de l'attribut `src` de l'élément `IMG` (ou `acc.src`, puisque cet élément se nomme `acc`), soit donc celle située à l'URI `images/accueil.gif`.

Lorsque la souris se place sur l'icône, la valeur de `acc.src` est placée dans la variable `t` et est remplacée par la valeur de `acc.lowsrc`, soit l'URI qui mène à la seconde icône : `images/accueil2.gif`. L'agent utilisateur applique alors la modification et affiche la seconde icône.

Lorsque le curseur quitte l'icône, la valeur de `t` est placée dans `acc.src` et la première icône est à nouveau affichée.



Modèle Objet de Document (DOM)

Le Modèle Objet de Document (DOM), spécification du W3C (**http://xmlfr.org/w3c/TR/REC-DOM-Level-1/**), est une interface de programmation d'applications (API) pour documents HTML et XML. Il se divise en deux parties : DOM Core (ou noyau) et DOM HTML (les spécifications propres à HTML).



Le DOM définit la structure logique des documents ainsi que la façon dont vous accédez aux éléments et les manipulez. Avec le Modèle Objet de Document, les programmeurs peuvent construire des documents, naviguer dans leur structure, et ajouter, modifier ou supprimer des éléments ou du contenu. Tout ce qui se trouve dans un document HTML ou XML peut être accédé, changé, détruit ou ajouté en utilisant le Modèle Objet de Document, à quelques rares exceptions près. DOM est conçu pour être utilisé avec n'importe quel langage de programmation.

Dans DOM, les documents ont une structure logique comparable à une arborescence. Le nom « Modèle Objet de Document » a été choisi parce qu'il s'agit d'un « modèle objet » au sens traditionnel de la conception orientée objet : les documents sont modélisés en utilisant des objets, tandis que le modèle ne contient pas uniquement la structure du document mais aussi son comportement et celui des objets dont il est composé. En d'autres termes, les nœuds ne représentent pas une structure de données, mais des objets possédant une identité, un comportement et des propriétés. Le DOM contient également les événements associés aux différents éléments du document.

Par exemple, dans l'exemple ci-dessus, DOM contient l'élément HTML `A` ainsi que les événements associés `onMouseOver` et `onMouseOut`. Les attributs des éléments HTML nommés sont traités comme des propriétés de l'objet. Ainsi, vous accédez à l'attribut `src` de l'élément `IMG` nommé `acc` à l'aide de `acc.src`.

Une étude plus complète du DOM dépasse largement l'objectif de ce livre. Reportez-vous si nécessaire à la spécification du W3C, à l'adresse indiquée en haut de cet encadré.

10.4. Travail avec les scripts

Vous devez l'avoir compris, si insérer un script, interne ou externe, dans un document HTML ne présente guère de difficultés, c'est tout autre chose dès que l'on passe à la rédaction de ces scripts.

Les langages de script sont généralement réputés ne pas être très faciles à apprendre. Si vous souhaitez vous y attaquer, commencez par la syntaxe : c'est la grammaire fondamentale des langages informatiques, de programmation, de script ou hypertextes. Après avoir compris la syntaxe, passez au vocabulaire : les mots réservés et les noms. Par exemple, en quoi un `frame` JavaScript diffère-t-il d'un `FRAME` HTML ?

Vous pouvez apprendre un langage de script en utilisant la même méthode que celle dont vous vous êtes servi pour HTML : en achetant un ouvrage consacré au langage qui vous intéresse, en examinant des codes source dans votre navigateur Web, ainsi qu'en menant des recherches dans les listes de diffusion et les forums de discussion.

Sans vouloir plonger trop avant dans l'étude de ces langages de script, il existe une méthode simple et efficace : réutiliser du code.

Vous avez remarqué que, dans les pages HTML que nous avons créées, de nombreuses parties de code étaient répétitives : plutôt que de les saisir à nouveau, un simple copier-coller permet d'obtenir rapidement le résultat souhaité. Il en va de même pour les scripts : réutiliser vos fragments de code économise du temps et des efforts. Si une fonction ou une sous-routine fonctionne à la perfection, mieux vaut ne pas la réécrire dans chaque document, mais en faire un script externe appelé par plusieurs documents. En outre, de nombreux concepteurs indépendants partagent leur travail avec tous, encourageant le public à utiliser et à réutiliser à leur guise leurs scripts. Lorsque vous voulez accomplir quelque chose, il existe de fortes chances pour qu'un programmeur autre que vous y ait déjà pensé et ait même proposé son script gratuitement sur le Net.

Un dernier point : la sécurité des scripts. Les scripts sont globalement bien moins dangereux qu'on ne le laisse supposer. Plus exactement, les dangers qu'ils présentent sont le plus souvent dus à des failles de sécurité des navigateurs et des systèmes d'exploitation. Des mises à jour sont régulièrement diffusées pour pallier ces failles lorsqu'elles sont identifiées. Veillez à toujours rechercher au moins mensuellement les mises à jour de sécurité qui s'appliquent à votre navigateur et à votre système d'exploitation.

Nous allons maintenant examiner un peu plus en détail les principaux langages de script.

JavaScript

Le JavaScript est un langage qui permet d'écrire des scripts incorporés dans un document HTML. Historiquement, c'est le premier langage de script pour le Web qui a permis aux développeurs Web d'exécuter de petits programmes sur le navigateur Web plutôt que sur le serveur Web, d'où le terme de script côté client.

JavaScript a été mis au point par Netscape en 1995. À l'origine, il se nommait LiveScript et était destiné à fournir un langage de script simple au navigateur Netscape Navigator 2. Il a, à l'époque, été longtemps critiqué pour son manque de sécurité, son développement peu poussé et l'absence de messages d'erreur explicites rendant délicate son utilisation. Le 4 décembre 1995, suite à une association avec le constructeur Sun, Netscape rebaptise son langage JavaScript (un clin d'œil au langage Java développé par Sun). À la même époque, Microsoft met au point le langage JScript, un langage de script très similaire.

De ce fait, le langage JavaScript dans ses différentes versions (jusqu'à 2.0 pour JavaScript au sens strict, plus les versions de JScript) est fortement dépendant du navigateur appelant la page Web dans laquelle le script est incorporé (reportez-vous plus loin au tableau page 407). Ainsi, pour éviter toute dérive, un standard a été défini pour normaliser les langages de script. Il s'agit de l'ECMA-262, aussi appelé ECMAScript, créé par l'organisation du même nom (ECMA, *European Computer Manufactures Association*). Nous reviendrons par la suite sur ECMAScript.

Il ne nécessite toutefois pas de compilateur, contrairement au langage Java avec lequel il a longtemps été et reste parfois confondu : le code est directement écrit dans la page HTML. C'est un langage peu évolué qui ne permet aucune confidentialité au niveau des codes : ceux-ci restent visibles.

Vous avez déjà rencontré JavaScript dans ce chapitre. Ce livre n'a pas pour vocation de vous apprendre tout ce qu'il faudrait savoir à propos de JavaScript, mais vous allez cependant en découvrir la syntaxe principale et le vocabulaire de base. Cela est suffisant pour débuter avec JavaScript.

JavaScript permet de personnaliser l'apparence de documents HTML, d'ouvrir et de fermer des boîtes de dialogue surgissantes (*pop-up*), de manipuler des cookies, d'exécuter des calculs ainsi que de générer et de manipuler dynamiquement le contenu de la page HTML elle-même. JavaScript a recours au Modèle Objet de Document (DOM), cité dans la section précédente, pour modifier dynamiquement la page Web.

La syntaxe de JavaScript diffère de celle d'HTML, sans en devenir pour autant effrayante. La syntaxe de base du langage, largement inspirée de celle de C/C++, présente les caractéristiques suivantes :

- Les commandes sont séparées par un point-virgule (`;`).
- Les accolades `{}`définissent un bloc de commande et permettent de limiter la portée des variables.
- Un commentaire sur une ligne est précédé de `//`.
- Un commentaire sur plusieurs lignes est encadré par la balise d'ouverture `/*` et la balise de fermeture `*/`.

JavaScript est sensible à la casse. Cela signifie que si vous concevez une fonction nommée `Date`, vous ne pourrez l'appeler à l'aide de `date`, `DATE` ou `DaTE` : il faut impérativement l'appeler avec `Date`.

JavaScript est enfin un langage script orienté objets, ce qui signifie que vous pouvez créer des classes et des instances d'objets. JavaScript propose en standard un certain nombre de classes, que vous pourrez utiliser à votre aise pour créer vos programmes. Il sera donc possible d'avoir des syntaxes du type :

```
objet.methode(parametre);
```

L'utilité de JavaScript pour les auteurs de pages Web dynamiques tient en grande partie au recours à des fonctions. Un script JavaScript peut être une simple ligne de code permettant de capturer la saisie d'une portion de texte, ou de la date et de l'heure courantes. Une fois la page chargée, il n'existe aucun moyen de mettre à jour l'heure sans avoir recours à un élément `META Refresh` ou sans demander à l'utilisateur d'actualiser (ou de recharger) la page. Si cependant le script qui permet d'inclure l'heure est situé dans une fonction JavaScript, celle-ci peut être exécutée chaque fois que l'utilisateur accomplit une certaine action, comme déplacer la souris sur une image ou cliquer sur un bouton.

Voici un exemple classique de code JavaScript, adapté pour une meilleure conformité avec la spécification HTML 4.01 à partir d'un fichier disponible gratuitement sur l'excellent site de Je JavaScript.net (**www.jejavascript.net/date.php**). Ce script n'a qu'un but : afficher la date système.

Listing 10-1 : date.js

```
// PLF - http://www.jejavascript.net/
var datedujour, date, mois, mois1, jour, jour1, an;
datedujour = new Date();
jour = datedujour.getDay()
switch(jour){
    case 1 :
       jour1 ="Lundi"
```

```
        break;
        case 2 :
           jour1 ="Mardi"
        break;
        case 3 :
           jour1 ="Mercredi"
        break;
        case 4 :
           jour1 ="Jeudi"
        break;
        case 5 :
           jour1 ="Vendredi"
        break;
        case 6 :
           jour1 ="Samedi"
        break;
        case 0 :
           jour1 ="Dimanche"
        break;
        }
    date = datedujour.getDate()
    mois = datedujour.getMonth()
    switch(mois+1){
        case 1 :
           mois1 ="Janvier"
        break;
        case 2 :
           mois1 ="F&eacute;vrier"
        break;
        case 3 :
           mois1 ="Mars"
        break;
        case 4 :
           mois1 ="Avril"
        break;
        case 5 :
           mois1 ="Mai"
        break;
        case 6 :
           mois1 ="Juin"
        break;
        case 7 :
           mois1 ="Juillet"
        break;
        case 8 :
           mois1 ="Ao&ucirc;t"
        break;
        case 9 :
           mois1 ="Septembre"
        break;
        case 10 :
           mois1 ="Octobre"
```

```
60         break;
61         case 11 :
62            mois1 ="Novembre"
63         break;
64         case 12 :
65            mois1 ="D&eacute;cembre"
66         break;
67         }
68     an = datedujour.getFullYear()
```

Les lignes ont été numérotées uniquement pour faciliter l'interprétation ultérieure.

Le fichier tel qu'il est présenté est un fichier externe, que vous devez appeler comme suit dans votre élément `HEAD` :

```
<SCRIPT type="text/javascript" src="date.js">
```

Ce code JavaScript calcule la date système. Comment fonctionne-t-il ?

1. Vous définissez dans la ligne 2 plusieurs variables : `datedujour`, `date`, `mois`, `mois1`, `jour`, `jour1` et `an`.
2. Dans la ligne 3, vous affectez à `datedujour` la valeur renvoyée par la méthode globale `Date()`.
3. Dans la ligne 4, vous affectez à la variable `jour` la valeur renvoyée par la méthode `datedujour.getDay()`. Celle-ci renvoie un nombre qui correspond au jour de la semaine.
4. Les lignes 5 à 27 sont une instruction conditionnelle qui affecte à la variable `jour1` le nom du jour de la semaine en lettres.
5. Dans la ligne 28, vous affectez à la variable `date` la valeur renvoyée par la méthode `datedujour.getDate()`. Celle-ci renvoie un nombre qui correspond au jour du mois.
6. Dans la ligne 29, vous affectez à la variable `mois` la valeur renvoyée par la méthode `datedujour.getMonth()`. Celle-ci renvoie un nombre qui correspond au mois. Les lignes 30 à 67 sont une instruction conditionnelle qui affecte à la variable `mois1` le nom du mois en lettres.
7. Dans la ligne 68, vous affectez à la variable `an` la valeur renvoyée par la méthode `datedujour.getFullYear()`. Celle-ci renvoie un nombre qui correspond au jour du mois.

Nous avons défini des variables, mais ne les avons pas encore employées : cela respecte le principe de la séparation des logiques de

travail et de présentation. Pour afficher la date du jour, placez ce qui suit dans un élément SCRIPT dans le corps de votre document HTML :

```
<SCRIPT type="text/javascript">
document.write("<DIV style='text-align: center;
  font-weight: bold; font-size: +2");
document.write(jour1, " ");
document.write(date, " ");
document.write(mois1 , " ");
document.write(an, " ");
document.write("</DIV>");
</SCRIPT
```

Cela écrit dans le document la date système, au sein d'un élément DIV à style défini.

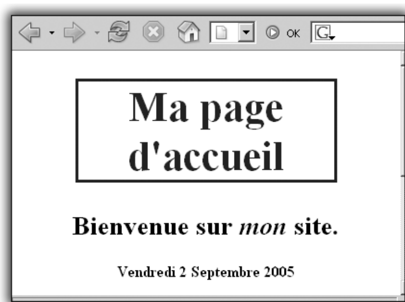


Figure 10.9 :
Affichage de la date système à l'aide d'un script

Ce script n'est qu'un exemple. Il pourrait être écrit de bien d'autres façons pour parvenir au même résultat. Par exemple, JavaScript permet l'utilisation de matrices dans vos scripts. Celles-ci sont définies à l'aide de crochets []. Vous pourriez ainsi employer des matrices au lieu d'instructions conditionnelles. Le programme deviendrait alors :

Listing 10-2 : date2.js

```
// PLF - http://www.jejavascript.net/ Modifs : F. Lemainque
var datedujour, date, mois, mois1, jour, jour1, an;
nomMois = new Array("Janvier","F&eacute;vrier","Mars",
  "Avril","Mai","Juin","Juillet","Ao&ucirc;t",
  "Septembre","Octobre","Novembre","D&eacute;cembre");
nomJours = new Array("Dimanche","Lundi","Mardi",
  "Mercredi","Jeudi","Vendredi","Samedi");
datedujour = new Date();
jour = datedujour.getDay()
jour1 = nomjours[jour]
date = datedujour.getDate()
mois = datedujour.getMonth()
mois1 = nomMois[mois]
an = datedujour.getFullYear()
```

Le fichier est nettement plus court : 11 lignes au lieu de 68 ! Il pourrait d'ailleurs être rendu encore plus compact, mais nous laissons cela à votre sagacité. Examiné dans votre navigateur, il procure des résultats identiques à la version précédente.

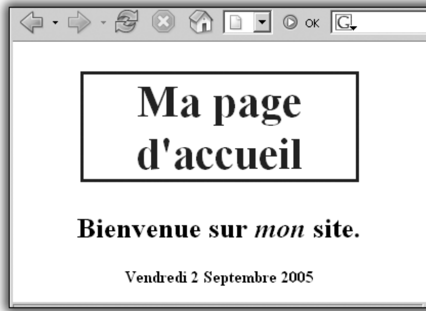


Figure 10.10 : *Autre script d'affichage de la date système*



Cadres cibles et JavaScript

JavaScript utilise des cadres cibles et des fenêtres nommées pour contrôler le contenu d'un document Web. Il peut vous jouer des tours quand vous ciblez à partir d'un cadre possédant une cible de base (élément `BASE`). Selon le navigateur, JavaScript peut employer la cible de base au lieu de la cible spécifiée dans le script. Vous pouvez contourner ceci en retirant l'élément `BASE` de votre page de cadres.

Comme tout bon langage, JavaScript possède un « vocabulaire » constitué de mots individuels qui, lorsqu'ils sont combinés en fonctions, ordonnent au navigateur Web d'accomplir quelque chose *via* le moteur de script. Vous trouverez certains des mots de ce vocabulaire dans le tableau page 406.

Très succinctement, comme nous l'avons signalé plus haut, vous pouvez décomposer le langage JavaScript en *objets*, *méthodes* et *événements*. L'ensemble d'*objets* disponibles pour le langage de script dépend du navigateur utilisé. Il s'agit du Modèle Objet de Document (*Document Object Model*, ou DOM), déjà abordé.

Les *méthodes* définissent les actions de JavaScript. Dans l'exemple précédent,

```
document.write(jour1, " ");
```

a recours à la méthode `write()` pour insérer du texte dans l'objet `document`, et le faire ainsi apparaître dans votre navigateur Web.

Enfin, les *événements* permettent d'appeler les fonctions. Si les fonctions sont un ensemble d'instructions exécutées chaque fois que vous prononcez le mot magique, alors l'événement est ce mot magique. Si le script contient une fonction nommée `OuvreFenetre`, vous pourriez appeler celle-ci à l'aide d'un événement comme :

```
<BODY onLoad="OuvreFenetre ()">,
```

Cela exécute le script. `onLoad` est l'événement qui indique au navigateur quand appeler la fonction et exécuter le script. Voici un exemple :

```
<SCRIPT LANGUAGE="javascript">
<!--
function OuvreFenetre(){
remote = window.open("","remotewin",width=125,height=175);
remote.location.href = "fenetre.html";
if (remote.opener == null) remote.opener = window;
remote.opener.name = "opener ";
}
//-->
</SCRIPT>
</HEAD>
<BODY onLoad ="OuvreFenetre()" bgcolor="#FFFFFF">
```

Si vous placez ce script dans une page Web et ouvrez celle-ci avec votre navigateur, vous pourriez obtenir l'avertissement présenté dans la figure suivante : ici les fenêtres surgissantes (*popup*) sont désactivées et Firefox vous prévient de la tentative.



Figure 10.11 :
Les fenêtres surgissantes sont désactivées

En autorisant les fenêtres surgissantes pour ce site et en rafraîchissant votre document dans le navigateur, vous voyez apparaître une nouvelle fenêtre.

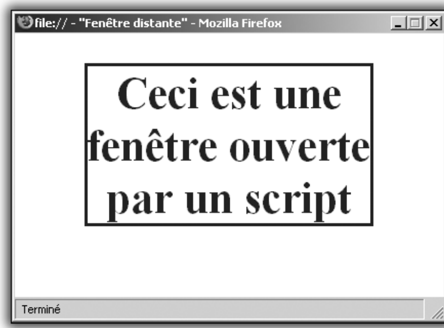


Figure 10.12 : *Apparition d'une fenêtre surgissante*

Globalement, les fenêtres surgissantes sont plus souvent pénibles qu'utiles : mieux vaut les laisser désactivées et ne les activer que lorsque cela est réellement nécessaire.

Le vocabulaire JavaScript contient bien d'autres mots, en sus des objets, méthodes et événements présentés ici : par exemple, les *matrices*, les *instructions* et les *opérateurs*. Une matrice est une collection d'objets ou de variables regroupés sous un même nom de variable. Nous en avons employé dans le fichier *date2.js* :

```
nomMois = new Array(
      "Janvier","F&eacute;vrier","Mars","Avril","Mai",
      "Juin","Juillet","Août","Septembre","Octobre",
      "Novembre","D&eacute;cembre");
```

Certaines instructions utilisent des mots réservés, comme `if`, `else if` et `switch`. Nous avions employé cette dernière instruction dans le fichier *date2.js* :

```
switch(jour){
… instructions case …
}
```

Chacune de ces instructions possède un sens précis pour JavaScript. C'est ainsi que `switch` signifie « branchez-vous sur l'instruction `case` qui correspond à la valeur du paramètre et accomplissez l'instruction mentionnée, puis sortez de l'instruction `switch` ».

Il existe également de nombreux opérateurs, les plus courants servant pour des opérations mathématiques ou comme opérateurs de comparaison, par exemple dans une instruction `if`.

Les objets documents, comme `document.cookie`, appartiennent aussi bien à JavaScript qu'au Modèle Objet de Document (DOM). JavaScript possède plusieurs objets intégrés, comme l'objet `cookie`, qui peuvent être utilisés automatiquement comme parties de l'objet document. Les mises en œuvre Netscape et Microsoft du DOM sont toutefois longtemps restées incompatibles, rendant difficile la vie du concepteur Web.

Le tableau suivant présente quelques exemples d'objets ou propriétés document, précieux en JavaScript.

Tableau 10.2 : *Propriétés de document JavaScript*

Nom de la propriété	Exemple	But	Équivalent HTML
alinkColor	document .alinkColor .#CC0000	Définit la couleur des liens sélectionnés	<BODY alink ="#CC0000">
bgColor	document .bgColor.#FFFFFF	Définit la couleur d'arrière-plan	<BODY bgcolor ="FFFFFF">
Cookie	document .cookie	Obtient la valeur de tout cookie associé à ce document	
fgColor	document .fgColor.#000000	Définit la couleur du texte	<BODY text ="#000000">
lastModified	document .lastModified	Obtient la date de dernière modification du document actif	
linkColor	document .linkColor .#0000CC	Définit la couleur des liens	<BODY link ="#0000CC">
Location	document .location	Obtient l'URL du document	<!–#echo var ="DOCUMENT_URI">
Referrer	document .referrer	Lit l'URL du document cible (la page Web liée au document actif)	
Title	document.title	Récupère le titre (l'élément TITLE) du document actif	
URL		Voir location	

Tableau 10.2 : *Propriétés de document JavaScript*			
Nom de la propriété	**Exemple**	**But**	**Équivalent HTML**
`vlinkColor`	`document .vlinkColor .#CC00CC`	Définit la couleur des liens visités	`<BODY vlink ="#CC00CC">`

Nous avons dit que le rendu et le bon fonctionnement de JavaScript dépendaient largement du navigateur employé.

Le tableau suivant présente les versions de JavaScript prises en charge par les principaux navigateurs.

Tableau 10.3 : *Versions JavaScript et navigateurs*	
Version de JavaScript	**Navigateurs**
JavaScript 1.0	Netscape Navigator 2.0, Internet Explorer 3.0, Opera, Mozilla
JavaScript 1.1	Netscape Navigator 3.0, Opera, Mozilla
JavaScript 1.2	Netscape Navigator 4.0/4.05, Internet Explorer 4.0, Opera, Mozilla
JavaScript 1.3	Netscape Navigator 4.06, Internet Explorer 5.0, Opera, Mozilla
JavaScript 1.4	Netscape Navigator 6.0, Internet Explorer 6.0, Opera, Mozilla
JavaScript 1.5	Netscape Navigator 6.0, Mozilla
JavaScript 1.6	Firefox 1.5/Gecko 1.8
JavaScript 1.7	Firefox 2/Gecko 1.8.1
JavaScript 1.8	Firefox 3/Gecko 1.9 (à paraître).

Si vous devez écrire plusieurs versions d'un programme, afin de tenir compte des agents utilisateurs employés par vos visiteurs, il serait intéressant de pouvoir déterminer le type d'agent utilisateur employé. Cela est possible : nous l'avons fait, quoique de façon très primitive, dans la dernière version du fichier *accueil.html* :

```
...
<SCRIPT type="text/javascript">
<!--
```

```
document.write("<P>Vous utilisez un navigateur "
   + navigator.appName + "</P>");
 -->
</SCRIPT>
...
```

Il est possible de faire bien mieux, grâce aux propriétés de l'objet `navigator` de JavaScript.

1 Ouvrez *accueil.html* dans le Bloc-Notes.
2 Enregistrez le fichier sous le nom *accueil_3_10_2.html*.
3 Modifiez le numéro de version :

```
<META name="version" content="3.10.3">
```

4 Déplacez l'élément `SCRIPT` par couper-coller juste avant l'élément `HR` du bas de la page.
5 Insérez après la ligne de commentaire de l'élément `SCRIPT` le code suivant, en remplacement de la ligne actuelle :

```
<SCRIPT type="text/JavaScript">
<!--
document.write("<TABLE width='70%'>");
document.write("<TR>");
document.write("<TD>Nom et version de votre navigateur
  </TD><TD>");
document.write(navigator.appName + " " + navigator
✂ .appVersion);
document.write("</TD></TR>");
document.write("<TR>");
document.write("<TD>Informations d&eacute;taillées sur
             votre agent utilisateur</TD><TD>");
document.write(navigator.userAgent);
document.write("</TD></TR>");
document.write("</TABLE>");
// -->
</SCRIPT>
```

6 Enregistrez votre fichier sous le nom *accueil.html*, puis ouvrez *index.html* dans un navigateur. Le résultat peut vous surprendre… (voir Figure 10.13, 10.14, 10.15).

Cela est obtenu à l'aide des précieuses propriétés de l'élément `navigator`, présentées dans le tableau suivant.

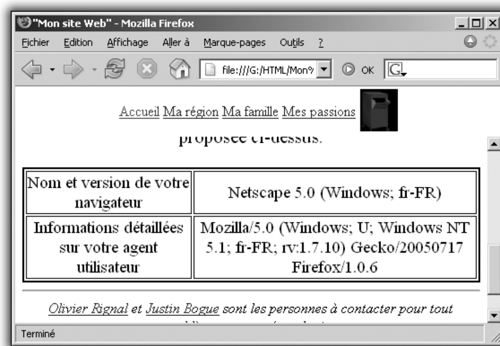


Figure 10.13 : *Affichage détaillé du type de navigateur et du système d'exploitation à l'aide d'un script JavaScript, respectivement pour Firefox, …*

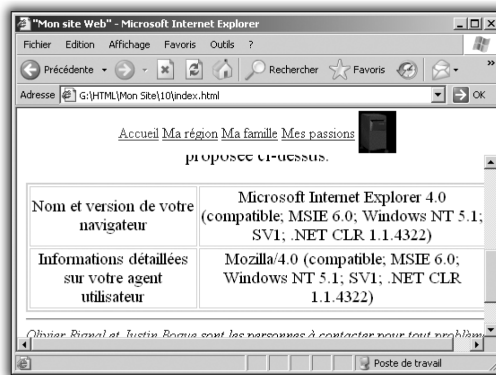


Figure 10.14 : *… pour Internet Explorer*

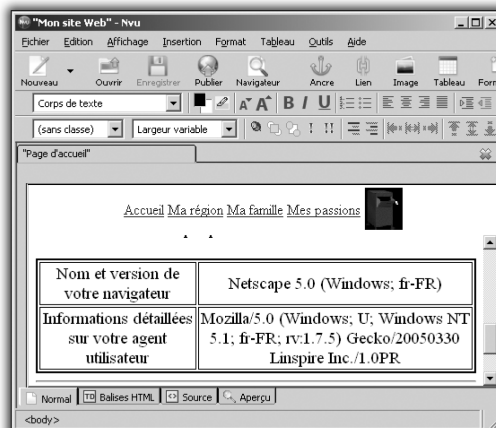


Figure 10.15 : *… pour Nvu.*

Tableau 10.4 : *Propriétés JavaScript de l'objet navigateur (navigator)*		
Propriété	**Description**	**Exemple**
`appCodeName`	Renvoie le nom de code du navigateur, le plus souvent Mozilla, le moteur utilisé par la plupart des navigateurs (Internet Explorer, Netscape, mais aussi beaucoup de navigateurs sous Unix.). Cette valeur sera différente si le navigateur du client est fondé sur un autre moteur, comme Opera.	`Mozilla`
`appName`	Renvoie le nom du navigateur (la plupart du temps la marque). Cette propriété est utile pour différencier les navigateurs de Netscape et de Microsoft.	`Netscape`
`appVersion`	Renvoie la version du navigateur sous la forme suivante : *numéro de version (système d'exploitation): nationalité)*. Elle est utile pour connaître le système d'exploitation de l'utilisateur, mais elle permet surtout, en association avec la propriété `navigator.appName`, de connaître les fonctionnalités prises en charge par le navigateur de votre visiteur.	`5.0 (Windows, fr-FR)`
`language`	Renvoie une chaîne de caractères qui correspond à la langue utilisée par le navigateur du client. Cette propriété n'est comprise que par les navigateurs prenant en charge les versions 1.2 et supérieures de JavaScript.	`fr-FR`
`mimeTypes`	Renvoie un tableau des types MIME pris en charge par le navigateur, c'est-à-dire les types de fichiers enregistrés.	`application/x-shockwave-flash, application/futuresplash, application/x-java-vm, etc.`
`platform`	Renvoie une chaîne de caractères qui indique le système d'exploitation du client. Cette propriété n'est comprise que par les navigateurs qui prennent en charge les versions 1.2 et supérieures de JavaScript.	`Win32`
`plugins`	Renvoie la liste des plug-in (modules complémentaires) installés sur la machine cliente.	`npnul32.dll, npswf32.dll, NPOJI610.dll, NPJava11.dll, etc.`

Tableau 10.4 : *Propriétés JavaScript de l'objet navigateur (navigator)*		
Propriété	**Description**	**Exemple**
`userAgent`	Renvoie la chaîne de caractères qui comprend toutes les informations sur l'agent utilisateur du client. Les propriétés ci-dessus offrent un moyen pratique de récupérer une partie de cette information.	`Mozilla/5.0 (Windows, U, Windows NT 5.1, fr-FR, rv:1.7.8), Gecko/20050511, Firefox/1.0.4`

Puisque vous pouvez déterminer le type de l'agent utilisateur du client, vous êtes en mesure de créer un site réellement dynamique, qui oriente le visiteur vers des pages conçues spécifiquement pour l'agent utilisateur dont il se sert. Par exemple, en remplaçant dans la barre de navigation

```
<TD><A href="accueil.html" target=princ>Accueil</A></TD>
```

par ce qui suit :

```
<script type="text/javascript">
<!--
Navigateur = navigator.appName;
if (Navigateur == 'Netscape')
{
document.write("<TD><A href='region_nets.html' ");
}
if (Navigateur == 'Microsoft Internet Explorer')
{
document.write("<TD><A href='region_IE.html' ");
}
document.writeln("target=princ>Ma r&eacute;gion</A></TD>"
//-->
</script>
```

Selon le navigateur, la page affichée diffère. Vous pouvez ainsi tirer profit des caractéristiques propres à chaque navigateur.



NOSCRIPT

Si vous procédez ainsi, vous devriez en pratique placer la ligne remplacée dans un élément NOSCRIPT, cela afin de prendre en compte les navigateurs ne reconnaissant pas les scripts, en ciblant éventuellement une page dépourvue de tout script :

```
<NOSCRIPT>
<TD><A href="region_sanscript.html" target=princ>
  Ma r&eacute;gion</A></TD>
</NOSCRIPT>
```

JavaScript est un des composants essentiels de la technologie AJAX (Asynchronous Javascript And XML). Les applications AJAX ont généralement recours à l'objet `XMLHTTPRequest` (XHR) pour envoyer une requête à un script serveur et analyser dynamiquement les résultats de ce dernier via DOM. Internet Explorer a été le premier à proposer ce composant, sous forme d'un ActiveX, dès la fin des années 90. Ce n'est qu'en 2002 que les développeurs commencent massivement à l'utiliser, vu son adoption dans FireFox. Les versions d'Internet Explorer antérieures à la 7 ne géraient pas l'objet XHR tels que décrit dans les standards du W3C mais proposaient un contrôle ActiveX équivalent (à partir de la version 5), ce qui impose des fourches dans le code.

AJAX est une des technologies phares du mouvement Web 2.0 qui définit les interfaces riches permettant à l'internaute une plus grande interactivité avec la page web.

ECMAScript

Comme nous l'avons déjà signalé, ECMAScript est une tentative de normalisation du noyau du langage JavaScript et de ses dérivés : sa syntaxe, ses mots-clés et ses composants natifs. La troisième édition du standard ECMA-262 a été publiée en décembre 1999 (**www.ecma-international.org/publications/standards/Ecma-262.htm**), mais la quatrième version est en développement (**http://www.ecmascript.org/es4/spec/overview.pdf**).C'est devenu un standard international (ECMA-262 et ISO-10262), ce qui en fait une des rares technologies Internet « non protocole » à avoir obtenu une normalisation d'un organisme officiel : même HTML n'a jamais dépassé le stade de Recommandation W3C.

Le standard ECMAScript définit une syntaxe de programmation proche du langage C. Si vous connaissez déjà ce langage, vous serez rapidement opérationnel.

Si chaque navigateur possède son propre interpréteur de JavaScript, et est donc susceptible d'être incomplet et de ne pas comprendre tout ce que vous aurez pu coder, les dernières versions des navigateurs modernes prennent en charge dans sa quasi-intégralité le standard ECMAScript. C'est pourquoi il vaut toujours mieux se reporter à ce standard lorsque vous écrivez un programme en JavaScript.

Vous ne pouvez identifier un script écrit en ECMAScript. En effet, il est déclaré dans du code HTML comme étant de type JavaScript :

```
<SCRIPT type ="text/JavaScript">
```

En pratique donc, les termes JavaScript et ECMAScript sont pratiquement synonymes, alors qu'ils peuvent recouvrir des différences notables.

VBScript

VBScript est un langage de script propre à Microsoft, dérivé du langage de programmation Visual Basic. VBScript n'était pas prévu initialement pour devenir un langage de script pour Internet. Il est destiné aux utilisateurs peu techniques, en utilisant largement la syntaxe Visual Basic. Il est de ce fait parfois plus pratique pour des utilisateurs occasionnels. L'expérience prouve qu'il suffit d'examiner quelques exemples de n'importe quel langage de script pour s'habituer à sa syntaxe.

VBScript reste très proche de JavaScript, même s'il emploie une syntaxe de ponctuation légèrement différente. Les commentaires débutent par une apostrophe simple, tandis que certains opérateurs ont recours à un symbole différent : par exemple `&` au lieu de `+`. VBScript n'est pas sensible à la casse : `document.write`, `document.Write` et `Document.Write` sont strictement équivalents.

Comme pour tout script, votre code doit être inséré dans un élément `SCRIPT`, mais avec cette fois, comme valeur de l'attribut `type`, la chaîne `text/vbscript`. Une ligne de commentaire permet de le masquer vis-à-vis des anciens navigateurs. Vous pouvez (et devriez) également avoir recours à un élément `<NOSCRIPT>` afin d'afficher du texte ou un lien pour les utilisateurs ne disposant pas de la reconnaissance de VBScript.

Voici un exemple de script VBScript. Il se borne à afficher la date système, exactement comme les fichiers JavaScript `date.js` et `date2.js` vus précédemment.

```
<SCRIPT type="text/vbscript">
<!--
document.write(Day(Now) & " " & MonthName(Month(Now), False)
 & " " & Year(Now))
' -->
</SCRIPT>
```

Hélas ! Comme le montrent les copies d'écran des figures suivantes, si cela fonctionne parfaitement avec Internet Explorer, ce script reste sans effet sous Firefox.



Figure 10.16 : *Test de ce script avec Internet Explorer*



Figure 10.17 : *Test de ce script avec Firefox.*

Il est toutefois une chose que VBScript gère bien mieux que JavaScript : ActiveX. Cela n'a rien d'étonnant, car ces deux technologies sont signées Microsoft. Un site conçu à l'aide de VBScript peut exploiter n'importe quel dispositif d'Internet Explorer : c'est le seul navigateur à reconnaître parfaitement à la fois VBScript et ActiveX. Bien évidemment, il est à peine nécessaire de vous rappeler la nécessité de prévoir un site alternatif pour les non-utilisateurs de MSIE.

Si VBScript présente les mêmes risques globaux de sécurité que n'importe quel script, ce n'est pas le cas d'ActiveX : Internet Explorer vous préviendra toutefois en principe des risques de certaines actions, ce qui vous permet de contrôler l'innocuité d'un programme avant son exécution.

Traitement de formulaires à l'aide de scripts

Vous avez vu, dans des exemples précédents, comment valider des entrées d'un formulaire à l'aide d'un script avant d'en autoriser la soumission.

Les scripts peuvent également agir (et agissent généralement) après la soumission. Lors de la conception de votre formulaire, vous devez spécifier une `action` dans l'élément `FORM`. Cette `action` détermine l'endroit où sont envoyées les données et leur traitement.

Il y a communément deux façons de traiter les données d'un formulaire. L'action la plus facile à utiliser est un formulaire d'envoi de courrier électronique :

```
<FORM action=”mailto:mon_nomph@monFAI.fr" method="POST">
```

Les données sont envoyées à mon adresse électronique sous une forme relativement illisible en raison du type d'encodage URL, qui a inséré des caractères supplémentaires avant la transmission au serveur ou l'envoi par courrier électronique. Vous avez alors besoin d'un moyen de décoder ces données. Il existe un grand nombre de programmes, dont de nombreux disponibles gratuitement (uudecode.com, Sperical UUDecode, etc.), qui permettent de décoder aisément ces données. Essayez simplement une recherche Google avec comme mots-clés « `uudecode + freeware` » !

L'autre façon de traiter les données consiste à employer un script CGI (*Common Gateway Interface*). Un script CGI est un script côté serveur. Les scripts CGI peuvent quasiment tout faire avec des données de formulaire : décoder les résultats et les envoyer à une adresse électronique, les ajouter à une base de données ou un fichier, chercher des informations, modifier un profil d'utilisateur, créer ou envoyer un fichier, effacer un fichier ou traiter les données comme bon vous semble.

Plusieurs scripts CGI gratuits sont disponibles sur Internet. La facilité d'emploi de ces scripts peut cependant varier. Ils sont généralement rédigés par des indépendants. Certains scripts ne sont gratuits que pour des institutions caritatives et éducatives, tous les autres devant payer. En général, un bon script CGI coûte entre 100 et 300 €.

Les programmes exécutés par CGI sont toujours écrits en langages de programmation individuels. La plupart des scripts CGI sont écrits en Perl, C ou C++, même si tout langage de programmation installé sur le

serveur peut être utilisé. C et C++ sont des choix fréquents, car de nombreux programmeurs sont familiarisés avec ce genre de langages. Malheureusement, les programmes C doivent être compilés dans quelque chose que l'ordinateur puisse comprendre. Perl est un langage très répandu pour les CGI, car il est petit, rapide et facile à apprendre et à utiliser. Contrairement à C, les scripts Perl n'ont pas besoin d'être compilés pour être exécutés. Enfin, un grand nombre de bibliothèques Perl ont été créées spécifiquement pour gérer les applications CGI.

Attention, les scripts CGI posent toutefois de nombreux problèmes :

- Comme ils sont installés côté serveur, ils en consomment les ressources. Cela peut ne plaire que très moyennement à votre hébergeur.
- Ils peuvent présenter de sérieux risques sécuritaires, soit volontairement, soit plus fréquemment parce qu'ils sont mal écrits : un autre motif qui les rend suspects aux yeux des hébergeurs !

Pour ces raisons, nombreux sont les hébergeurs qui soit refusent totalement l'installation de tout script CGI, soit, plus rarement, offrent eux-mêmes une palette de scripts sévèrement contrôlés auxquels vous devez obligatoirement vous limiter.

Cookies

Un cookie est un petit fragment d'information qui reste stocké sur votre ordinateur après la clôture de la connexion. HTTP est un protocole dit « sans état », ou « sans mémoire » : chaque requête individuelle envoyée au serveur Web est indépendante des précédentes. Autrement dit, le serveur ne dispose d'aucun moyen de savoir que vous venez de consulter la page *Ma région* alors que vous accédez à la page *Ma famille*.

Un fichier de cookie est un fichier en texte brut renfermant des cookies. Netscape rassemble les cookies dans votre dossier *Profil utilisateur*. Chaque utilisateur possède son propre fichier de cookies, exactement comme il dispose de sa propre liste de signets.

Avec Internet Explorer, les cookies sont stockés dans le dossier *C:\Windows\Cookie* du disque dur. Chaque cookie constitue un fichier individuel : en raison de la taille minimale occupée par un fichier, cela

peut amener à consommer un espace disque non négligeable. Il en va d'ailleurs de même pour la liste des favoris.

Les utilisateurs ne font généralement pas grand-chose avec leurs cookies, et ne sont d'ailleurs pas supposés y toucher. Même si de nombreux utilitaires de gestion des cookies, désormais souvent intégrés aux navigateurs, rendent ceux-ci accessibles à l'utilisateur moyen, le but réel des cookies est de fournir certaines informations au serveur Web, pour maintenir un état homogène lors de la navigation sur le Web.

Les cookies renferment parfois des informations sensibles, comme des noms d'utilisateurs ou des mots de passe : le genre de choses qu'il n'est guère souhaitable de voir rendues disponibles à tout autre site que celui pour lequel elles sont prévues. Rassurez-vous, un cookie ne peut être lu que par le serveur Web qui l'a généré. Si vous possédez un serveur Web à l'adresse **www.mondomaine.com** et que vous créez un cookie utilisateur contenant le nom et le numéro de téléphone de l'utilisateur, seuls les scripts s'exécutant sur **www.mondomaine.com** sont capables de lire ce cookie et d'en extraire le numéro de téléphone.

Un cookie peut servir à stocker des informations saisies par l'utilisateur dans un formulaire Web ou tout autre périphérique d'entrée en ligne. Il peut aussi mémoriser des comportements, comme les liens hypertextes suivis ou les images affichées. Cela permet au responsable du site de cibler sa publicité, d'évaluer l'efficacité des différents dispositifs de navigation et d'identifier les sujets qui retiennent l'attention des visiteurs.

Malgré leur potentiel et en dépit de certaines idées reçues, les cookies restent limités. Ils ne peuvent exécuter un programme situé sur votre disque dur, télécharger ou installer un virus, imposer à votre ordinateur d'envoyer un message électronique à votre insu, lire un élément non volontairement fourni par l'intermédiaire d'un formulaire Web ou, d'une façon plus générale, provoquer une quelconque modification de votre système. Un cookie stocke une information élémentaire dans un endroit précis de votre disque dur : le fichier *cookies.txt* avec Netscape, le dossier *C:\Windows\Cookies* avec Internet Explorer. C'est tout. L'information qu'il renferme ne peut venir que du serveur Web, à l'aide de données transmises *via* un formulaire Web ou tout autre dispositif de saisie en ligne.

En outre, un utilisateur peut choisir d'accepter ou de refuser les cookies. Par exemple, pour ce faire, avec Firefox, choisissez **Outils** > **Options**.

Sélectionnez **Vie privée** dans le volet de gauche, développez la section **Cookies** et décochez l'option *Autoriser les sites à créer des cookies*.

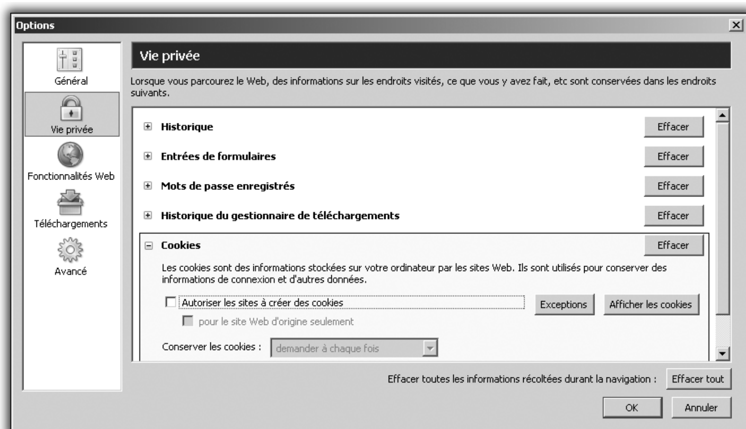


Figure 10.18 : *Contrôle des cookies avec Firefox*

Vous pouvez paramétrer Firefox afin qu'il vous demande si vous souhaitez accepter un cookie.



Figure 10.19 : *Invite d'acceptation de cookie de Firefox*

Avec Internet Explorer, choisissez **Outils** > **Options Internet**. Cliquez sur l'onglet **Confidentialité**, puis cliquez sur *Avancées*. Dans la fenêtre qui s'affiche, sélectionnez les options adéquates.

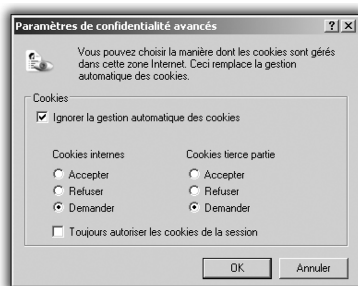


Figure 10.20 : *Contrôle des cookies avec Internet Explorer*

Vous pouvez créer un cookie à l'aide de code HTML pur, à l'aide d'un élément `META` et d'un en-tête http ou à l'aide de l'attribut `HTTP_EQUIV`.

```
<META HTTP-EQUIV="Set-Cookie" CONTENT="cookievalue=xxx;
expires=Saturday, 31-Dec-05 23:59:59 GMT; path=/">
```

Cela n'offre toutefois qu'un intérêt relatif, car tout y est codé « en dur », tandis que le bon fonctionnement n'est garanti qu'avec Netscape.



`HTTP–EQUIV`

Nous évoquons ici l'attribut `HTTP–EQUIV` d'un élément `META`. Un élément `META` doté d'un attribut `HTTP–EQUIV` équivaut à un en-tête HTTP. Il sert à contrôler l'action des navigateurs et peut permettre de modifier ou de préciser les informations fournies par les vrais en-têtes. Ces éléments devraient avoir un effet identique qu'ils soient sous cette forme ou qu'ils soient spécifiés comme en-têtes HTTP. Certains serveurs les traduisent automatiquement en vrais en-têtes HTTP. Les en-têtes HTTP sont généralement supplantés par les instructions HTML correspondantes et sont parfois interprétés de façons différentes par les serveurs. Nous avons donc préféré les laisser de côté. Reportez-vous pour plus d'informations à la spécification HTTP (RFC 2616, **www.ietf.org/rfc/rfc2616.txt**).

Mieux vaut donc avoir recours à un petit programme JavaScript, comme celui-ci :

Listing 10-3 : fonction Set_Cookie

```
function Set_Cookie( nom, valeur, expiration, pathname,
  mondomaine, secure )
{
// Définition de la date/heure exprimée en milliseconds
var today = new Date();
today.setTime( today.getTime() );

/*
Si la variable expiration est définie, calculer la date
d'expiration exacte. Elle est exprimée ici en jours
*/
if ( expiration )
{
expiration = expiration * 1000 * 60 * 60 * 24;
}
var date_expiration = new Date( today.getTime()
  + (expiration) );

document.cookie = nom + "=" +escape( valeur ) +
( ( expiration ) ? ";expiration="
```

```
  + date_expiration.toGMTString() : "" ) +
( ( pathname ) ? ";pathname=" + pathname : "" ) +
( ( mondomaine ) ? ";mondomaine=" + mondomaine : "" ) +
( ( secure ) ? ";secure" : "" );
}
```

Le cookie est prêt. Dans la plupart des cas, seuls les quatre premiers paramètres doivent être fixés (`nom`, `valeur`, `expiration` et `nom_path`). Si le cookie ne doit être disponible que dans un dossier, ajoutez comme variable `nom_path '/nom_dossier/'` pour indiquer au script de définir ce cookie uniquement pour ce chemin et non pour la totalité du domaine. En général toutefois, vous donnerez à la variable `path` la valeur `'/'`, la racine de votre site Web. `'mondomaine'` et `'secure'` n'ont généralement pas besoin d'être spécifiés, sauf dans des buts très précis.

Vous créez réellement le cookie en invoquant cette fonction :

```
Set_Cookie( 'moncookie', '9', 30, '/', '', '' );.
```

Cette ligne de code crée un cookie nommé `moncookie`, dont la valeur est `9` et la date d'expiration fixée à 30 jours de sa création, valides pour la totalité du site.

Veillez à toujours faire figurer le nombre correct de paramètres, les paramètres non spécifiés étant placés entre apostrophes vides. Un nombre incorrect de paramètres entraîne une erreur du code.

Créer un cookie est parfait, mais pouvoir le lire serait encore mieux. Voici comment procéder :

Listing 10-4 : fonction Get_Cookie

```
// cette fonction récupère le cookie s'il existe
function Get_Cookie( nom ) {

var debut = document.cookie.indexOf( nom + "=" );
var longueur = debut + nom.length + 1;
if ( ( !debut ) &&
( name != document.cookie.substring( 0, nom.length ) ) )
{
return null;
}
if ( debut == -1 ) return null;
var fin = document.cookie.indexOf( ";", len );
if ( fin == -1 ) fin = document.cookie.length;
return unescape( document.cookie.substring( longueur, fin ) );
}
```

Vous pouvez ensuite employer cette fonction comme suit :

```
if ( Get_Cookie( 'moncookie' ) ) faire quelque chose
```

« Parfait », direz-vous, mais à quoi cela peut-il donc servir ? Eh bien, que penseriez-vous de pouvoir accueillir de façon personnalisée les visiteurs de votre site, en indiquant le nombre de leurs visites sur le site ?

Pour ce faire, vous allez écrire un petit (!) programme JavaScript. Attention, si vous le saisissez réellement, faites-le sans les numéros de ligne !

Listing 10-5 : passage.js

```
01 // Passage.js
02 var username = get_Cookie('username');
03 if (username == null ) {
04   username = prompt('Merci de bien vouloir taper votre
        nom ou votre pseudo ',"");
05   if (username == null) {
06     username="vous";
07   } else {
08     pathname = location.pathname;
09     myDomain = pathname.substring(0,
          pathname.lastIndexOf('/')) +'/';
10     // définir la date d'expiration à un an .
11     var largeExpDate = new Date ();
12     largeExpDate.setTime(largeExpDate.getTime()
          + (365 * 24 * 3600 * 1000));
13   }
14   set_Cookie('username',username,largeExpDate,myDomain);
15 }
16
```

La ligne 2 définit une variable `username`, dont la valeur est spécifiée comme égale à celle du champ *username* de la fonction `get_Cookie`. Celle-ci n'est pas encore définie, mais il est classique dans un programme JavaScript de placer les définitions des fonctions à la fin du programme.

Si le nom d'utilisateur est vide (test de la ligne 3), la ligne 4 ouvre une boîte de dialogue invitant l'utilisateur à saisir son nom ou son pseudo. Comme l'utilisateur peut parfaitement refuser de saisir son nom en refermant la boîte d'invite, les lignes 5 à 6 gèrent ce cas en attribuant à la variable `username` la valeur `vous`, puis saute à la ligne 13, qui marque la fin de cette instruction conditionnelle.

Si l'utilisateur a renseigné son nom ou pseudo, les lignes 7 et 8 spécifient respectivement les valeurs des variables `pathname` et `monDomaine`. Les lignes 9 à 11 calculent la date d'expiration en ajoutant 365 jours à la date système. La ligne 12 crée un cookie à partir des différentes valeurs renseignées ou calculées. Enfin, les lignes 13 et 14 referment les deux instructions conditionnelles. Un cookie qui contient le nom de l'utilisateur est alors créé (ou recréé).

Voici maintenant les fonctions du programme. Vous reconnaîtrez : `get_Cookie` (lignes 25 à 40) et `set_Cookie` (lignes 42 à 55), un peu modifiées par rapport aux précédentes ; une fonction utilitaire, `get_cookieVal` (lignes 17 à 23, qui extrait une portion du cookie en fonction d'un paramètre de décalage (`offset`) ; et enfin la fonction `message` (lignes 57 à 60).

```
17 function get_CookieVal (offset)
18 {
19    var endstr = document.cookie.indexOf (";", offset);
20    if (endstr == -1)
21       endstr = document.cookie.length;
22    return unescape(document.cookie.substring(offset, endstr));
23 }
24
25 function get_Cookie (name)
26 {
27         var arg = name + "=";
28         var alen = arg.length;
29         var clen = document.cookie.length;
30         var i = 0;
31         while (i < clen) {
32                 var j = i + alen;
33                 if (document.cookie.substring(i, j) == arg)
34                         return get_CookieVal (j);
35                 i = document.cookie.indexOf(" ", i) + 1;
36                         if (i == 0)
37                                 break;
38                 }
39    return null;
40 }
41
42 function set_Cookie (name, value)
43 {
44         var argv = set_Cookie.arguments;
45         var argc = set_Cookie.arguments.length;
46         var expires = (argc > 2) ? argv[2] : null;
47         var path = (argc > 3) ? argv[3] : null;
48         var domain = (argc > 4) ? argv[4] : null;
49         var secure = (argc > 5) ? argv[5] : false;
50         document.cookie = name + "=" + escape (value) +
51           ((expires == null) ? "" : ("; expires=" +
```

```
52           expires.toGMTString())) + ((path == null) ? "" : (";
✂ path=" +
53           path)) + ((domain == null) ? "" : ("; domain=" +
✂ domain)) +
54           ((secure == true) ? "; secure" : "");
55 }
56
57 function message(){
58 document.write ("Bonjour "+ username + " ! Cela fait  " +
✂ visites +
59                " fois que vous visitez notre site");
60 }
61
```

Nous n'entrerons pas dans le détail de ces fonctions, car cela risquerait de nous emmener trop loin. Pour terminer le script, saisissez ce qui suit :

```
62 var expdate = new Date();
63 var visites;
64 // définir la date d'expiration à un an .
65 expdate.setTime(expdate.getTime() +  (24 * 60 * 60 * 1000 *
✂ 365));
66 if(!(visites = get_Cookie("visites")))
67   visites = 0;
68 visites++;
69 set_Cookie("visites", visites, expdate, "/", null, false);
70 username = username.toUpperCase();
71 message();
```

Vous définissez la date d'expiration et incrémentez la variable compteur `visites`, puis créez le cookie nommé `visites`. La ligne 79 convertit par sécurité la variable `username` en majuscules, afin d'éviter toute erreur de casse, tandis que la dernière ligne invoque la fonction `message`. Enregistrez ce fichier sous le nom *passage.js*.

1. Ouvrez maintenant dans le Bloc-Notes le fichier *accueil.html*.
2. Enregistrez-le sous le nom *accueil_3_10_3.html*, puis modifiez son numéro de version :

   ```
   <META name="version" content="3.10.4">
   ```

3. Pour invoquer votre programme JavaScript, placez ce qui suit juste après l'élément `H2` :

   ```
   <P><SCRIPT type="txt/JavaScript" src="passage.js">
   message();</SCRIPT>
   ```

4. Enregistrez votre fichier sous le nom *accueil.html*, puis ouvrez *index.html* dans votre navigateur.

Vous voyez d'abord apparaître la boîte de dialogue d'invite.

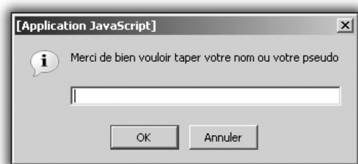


Figure 10.21 :
Invite de saisie du nom d'utilisateur

Saisissez (ou non) un nom, fermez la boîte de dialogue : un message de bienvenue s'affiche à l'écran.

Si vous cliquez sur le bouton **Rafraîchir** (ou **Actualiser**) de votre navigateur, le cookie `username` est lu, le compteur de visites est incrémenté et le message se modifie : il en sera de même à chaque visite.

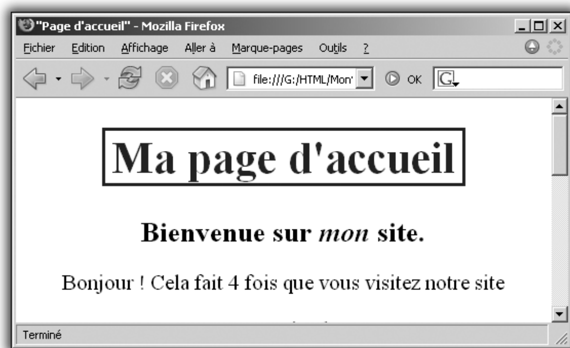


Figure 10.22 :
Le compteur de visites est incrémenté à chaque passage sur la page

À partir de ce script, il serait facile de modifier le message en fonction du nombre de visites ou d'après des noms d'utilisateurs spécifiques. Attention toutefois à ne pas placer ainsi des données confidentielles, car il est facile d'accéder au fichier *passage.js* et d'examiner son contenu.

Vous venez de découvrir la puissance des cookies, limitée toutefois par la possibilité qu'ont les utilisateurs de les refuser. Examinons maintenant un autre outil fréquemment employé, les applets.

10.5. Applets

Le lancement de Java par Sun Microsystems en 1995 a fait l'effet d'une bombe. Tout le monde voulait soudainement du Java sur son site. Celui-ci servait à résoudre des problèmes alors que cela n'était pas du tout son but initial.

La popularité de Java tenait à ce qu'il pouvait accomplir deux choses dont était jusque-là incapable le Web. Vous pouviez tout d'abord concevoir des applications compatibles quelle que soit la plate-forme considérée, exécutables sur le Web en temps réel. Il constituait ensuite un moyen dynamique de contrôle et de manipulation de données Web, toujours en temps réel.

L'HTML dynamique a très temporairement perturbé Java. Il permet lui aussi la manipulation en temps réel de contenus Web, tout en étant chargé plus rapidement. DHTML n'est toutefois encore que peu compatible entre les navigateurs, sans même évoquer des architectures différentes.

Les applets Java étant partiellement compilées, vous ne pouvez en examiner le code comme avec HTML, JavaScript ou VBScript. Cela est capital pour les développeurs Java désireux de protéger leur code. Le tableau suivant rappelle les principales différences entre JavaScript et Java.

Tableau 10.5 : *Différences entre JavaScript et Java*

JavaScript	Java
Langage interprété	Langage pseudo compilé (chargement d'une machine virtuelle)
Code intégré au HTML (éventuellement *via* un fichier externe)	Code (applet) à part du document HTML, appelé à partir de la page.
Langage peu typé	Langage fortement typé (déclaration du type de variable)
Liaisons dynamiques : les références des objets sont vérifiées au chargement	Liaisons statiques : les objets doivent exister au chargement (compilation)
Accessibilité du code	Confidentialité du code
Sûr : ne peut pas écrire sur le disque dur	Sûr : ne peut pas écrire sur le disque dur

Remarquez que si une applet Java (le programme) est préalablement compilée, une machine virtuelle permettant d'interpréter le pseudo code doit être chargée en mémoire (du côté du client) à chaque chargement de la page, d'où un important ralentissement pour les applets Java par rapport au JavaScript.

Les failles originelles de Java autorisaient la manipulation et la modification de fichiers sur votre disque dur. Sun a rapidement remédié au problème, mais comme les applets Java peuvent désormais demander et obtenir des droits d'accès supplémentaires, des applets provenant de sources indélicates sont susceptibles de contenir des virus et des programmes « douteux » : soyez prudents avec les applets Java. Mieux vaut généralement les désactiver dans votre navigateur.

Cette attitude étant conseillée, elle diminue l'intérêt des applets Java dans vos pages HTML.

HTML permet d'intégrer une applet dans votre page Web. Il ne s'agit toutefois pas obligatoirement d'une applet Java : ce peut être une applet Python, Perl, etc. L'inclusion d'une applet s'effectuait autrefois à l'aide de l'élément `APPLET`, désormais déconseillé au profit de l'élément `OBJECT`. Voici un exemple de code HTML d'insertion d'une applet à l'aide de l'élément `APPLET` :

```
<APPLET code="nomclasse.class" width="100" height="100"></APPLET>.
```

La valeur de l'attribut `code` est le nom de la classe Java qui sert de fichier principal de l'applet elle-même. Dans l'élément `APPLET`, vous pouvez placer un ou plusieurs paramètres dans des éléments `PARAM`. Ceux-ci indiquent à l'applet les variables et contrôles qu'elle doit utiliser. La plupart des bonnes applets que vous pouvez télécharger et utiliser gratuitement possèdent des commentaires relatifs aux variables et paramètres que vous pouvez et devriez utiliser.

Élément OBJECT

Désormais, vous devez préférer l'élément `OBJECT` :

```
<OBJECT codetype="application/java" classid="nomclasse.class"></OBJECT>
```

Le tableau suivant présente les principaux attributs de l'élément `OBJECT`.

Tableau 10.6 : *Attributs classiques de l'élément OBJECT*		
Attribut	**Valeur**	**Description**
`classid`	`uri`	Sert à spécifier la localisation de l'implémentation d'un objet *via* un URI. Peut être employé simultanément ou en alternative avec l'attribut `data`, selon le type de l'objet impliqué.

Tableau 10.6 : Attributs classiques de l'élément OBJECT		
Attribut	**Valeur**	**Description**
`codebase`	`uri`	Spécifie le chemin de base utilisé pour résoudre les URI relatifs spécifiés par les attributs `classid`, `data` et `archive`. Sa valeur par défaut est l'URI de base du document courant.
`codetype`	`type-de-contenu`	Spécifie le type de contenu des données attendues lors du chargement de l'objet spécifié par l'attribut `classid`. Facultatif, il est toutefois recommandé lorsque l'attribut `classid` est spécifié : il permet à l'agent utilisateur d'éviter le chargement d'informations de type non reconnu. Sa valeur par défaut est celle de l'attribut `type`.
`data`	`uri`	Peut être utilisé pour spécifier la localisation des données de l'objet (par exemple les données d'images) ou, plus généralement, la forme sérialisée d'un objet qui peut être utilisée pour recréer cet objet. Quand il est donné comme URI relatif, celui-ci devrait être interprété relativement à l'attribut `codebase`.
`type`	`type-de-contenu`	Spécifie le type de contenu des données spécifiées par l'attribut `data`. Facultatif, mais recommandé quand l'attribut `data` est spécifié : il permet à l'agent utilisateur d'éviter le chargement des informations à type de contenu non reconnu. Si la valeur de cet attribut diffère de l'en-tête HTTP `Content-Type` renvoyé par le serveur quand l'objet est ramené, c'est l'en-tête HTTP `Content-Type` qui a préséance.
`archive`	`liste-uri`	Sert à spécifier la liste des URI, séparés par des espaces, des archives contenant les ressources qui concernent l'objet. Cette liste peut inclure les ressources spécifiées par les attributs `classid` et `data`. Le préchargement des archives diminue généralement le temps de chargement des objets. Les archives spécifiées comme URI relatifs devraient être interprétées relativement à l'attribut `codebase`.
`declare`		Quand il est présent, cet attribut booléen fait de la définition de l'élément `OBJECT` courant seulement une déclaration. L'objet doit être instancié par la suite au travers d'une définition `OBJECT` qui se réfère à cette déclaration.
`standby`	`texte`	Spécifie le message que l'agent utilisateur peut restituer pendant le chargement de l'implémentation et des données de l'objet.

Voici un exemple d'insertion d'applet d'horloge analogique dans un document *via* l'élément OBJECT. L'applet, écrit dans le langage Python, ne requiert aucune données supplémentaires ni valeurs d'exécution. L'attribut classid spécifie la localisation de l'applet :

```
<P><OBJECT classid="http://www.miamachina.it/orologioanalogico.py">
</OBJECT>
```

Remarquez que l'horloge sera restituée dès l'interprétation de la déclaration de l'élément OBJECT par l'agent utilisateur. Il est possible de différer la restitution d'un objet en effectuant une déclaration préalable de cet objet.

Mieux vaut toujours ajouter à un tel élément un texte de remplacement, au cas où l'agent utilisateur ne puisse pas restituer l'horloge.

```
<P><OBJECT classid="http://www.miamachina.it/orologioanalogico.py">
Une horloge anim&eacute;e.
</OBJECT>
```

Une des grandes forces de l'élément OBJET est, comme nous l'avons vu dans le Chapitre 6 traitant des images, la possibilité de spécifier des restitutions alternatives de l'objet : chacune des déclarations des éléments OBJECT imbriqués peut spécifier un type de contenu alternatif. Si l'agent utilisateur ne peut pas restituer l'élément OBJECT le plus externe, il essaye d'en restituer le contenu, qui peut être un autre élément OBJECT, etc. Souvenez-vous de l'exemple alors présenté :

```
<TD rowspan="3">
<!-- Essayer d'abord un applet en Python -->
<OBJECT title="Coucher de soleil sur le lac de Linciel"
  classid="Linciel.py">
  <!-- Sinon, essayer l'animation MNG -->
  <OBJECT data="Linciel.mng" type="application/mng">
    <!-- Sinon, essayer l'image JPEG -->
    <OBJECT data="Images/coucher de soleil.jpg"
    type="image/gif" width="267" height="200"
    title="Un coucher de soleil sur le lac de Linciel">
    <!-- Sinon, le texte en dernier recours -->
    Coucher de soleil sur le lac de Linciel
    </OBJECT>
  </OBJECT>
</OBJECT>
</TD>
```

L'agent utilisateur essaye d'abord de restituer le premier élément OBJECT possible, dans l'ordre suivant :

1 Une applet écrite en langage Python.

2 Une animation *MNG* du coucher de soleil.
3 Une image *JPG* de celui-ci.
4 Un texte de remplacement.

La déclaration la plus externe spécifie une applet qui ne requiert ni données ni valeur initiales. La deuxième déclaration spécifie une animation *MNG*. Comme elle ne définit pas la localisation d'une implémentation pour traiter l'animation *MNG*, elle compte sur l'agent utilisateur pour la prendre en charge. La troisième déclaration spécifie la localisation d'un fichier *JPG* et fournit un texte de remplacement au cas où tous les autres mécanismes auraient échoué.

Les données à restituer peuvent être fournies de deux façons : en interne ou à partir d'une ressource externe. La première méthode est généralement plus rapide, mais ne convient pas pour une grande quantité de données.

Voici un exemple d'emploi de données internes dans l'élément `OBJECT` :

```
<P>
<OBJECT id="horloge1"
        classid="clsid:663C8FEF-1EF9-11CF-A3DB-080036F12502"
        data="data:application/x-oleobject;base64, ...des
        ✂ donn&eacute;es en base64...">
    Une horloge.
</OBJECT>
```

Élément PARAM

Pour initialiser un élément, vous avez recours à un ou plusieurs éléments `PARAM`. Cet élément ne possède qu'une balise d'ouverture.

Les éléments `PARAM` spécifient l'ensemble des valeurs nécessaires à l'exécution d'un objet. Un élément `OBJECT` (ou `APPLET`) peut posséder un nombre quelconque d'éléments `PARAM` dans n'importe quel ordre. Ils doivent toutefois être situés au début du contenu de l'élément `OBJECT` (ou `APPLET`).

La syntaxe des noms et de leurs valeurs est censée être comprise par l'implémentation de l'objet. La spécification ne définit pas les façons selon lesquelles les agents utilisateurs devraient ramener les couples nom/valeur ni interpréter les noms en doublon.

L'attribut name définit le nom d'un paramètre d'exécution censé être connu des objets insérés. La sensibilité à la casse de la propriété dépend de l'implémentation spécifique de l'objet.

L'attribut value spécifie la valeur d'un paramètre d'exécution spécifié par l'attribut name. Les valeurs de propriété n'ont aucune signification pour HTML : leur signification est déterminée par l'objet en question.

L'attribut valuetype spécifie le type de l'attribut value. Les valeurs possibles sont :

- data : c'est la valeur par défaut. Signifie que la valeur spécifiée par value sera évaluée puis transmise à l'implémentation de l'objet sous forme d'une chaîne.
- ref : la valeur spécifiée par l'attribut value est un URI qui désigne une ressource dans laquelle les valeurs d'exécution sont stockées. L'URI doit être transmise à l'objet tel quel (non résolu).
- object : la valeur spécifiée par l'attribut value est un identifiant qui se réfère à la déclaration d'un élément OBJECT situé dans le même document. L'identifiant doit être la valeur de l'attribut id de l'élément OBJECT déclaré.

L'attribut type spécifie le type de contenu de la ressource désignée par l'attribut value uniquement lorsque l'attribut valuetype possède comme valeur ref. Cet attribut indique donc à l'agent utilisateur le type des valeurs situées à l'URI désigné par l'attribut value.



OBJECT **et** EMBED

Microsoft Internet Explorer reconnaît l'élément OBJECT à partir de sa version 3.0, Netscape à partir de sa version 4. Ce dernier prenait également en charge l'élément EMBED qui lui était propre. Vous devriez désormais n'employer que OBJECT.

Voici un exemple d'emploi de ces deux éléments, qui tient compte du navigateur :

Listing 10-6 : Éléments OBJECT et EMBED

```
...
<OBJECT classid="clsid:numericID" codebase="../"
  width="largeur" height="hauteur" id="nom">
<PARAM name="src" value="nomfichier.dcr">
<PARAM name="bgcolor" value="couleurfond">
<PARAM name="palette" valeur="fond">
<!-- Netscape 2.0 et 3.0 ont recours à EMBED -->
```



```
<EMBED width="largeur" height="hauteur" src="nomfichier.dcr"
bgcolor="couleurfond"
pluginspage="http://www.macromedia.com/shockwave/download/"
start="true" alt="Shockwave">
<NOEMBED> Si vous ne disposez pas de Shockwave, vous pouvez
  le t&eacute;l&eacute;charger
<A HREF=http://www.macromedia.com/shockwave/download/> ici
</A>
pour voir cette animation.
</NOEMBED>
</OBJECT>
...
```

Un utilisateur qui possède un navigateur ne reconnaissant ni `OBJECT` ni `EMBED` voit le contenu de l'élément `NOEMBED`. Remarquez que les éléments `OBJECT` et `NOEMBED` doivent posséder une balise de fermeture, contrairement à `EMBED`. Désormais, vous devriez toutefois n'employer que `OBJECT`.

Déclaration et instanciation des objets

Nous avons signalé plus haut qu'il était possible de séparer la déclaration d'un objet de ses instanciations. Cela présente plusieurs avantages :

- Les données peuvent être ramenées du réseau par l'agent utilisateur une seule fois (lors de la déclaration) et réutilisées pour chaque instanciation.
- Il est possible d'instancier un objet à partir d'un autre endroit que celui de la déclaration de l'objet, par exemple à partir d'un lien.
- Il est possible de spécifier des objets comme étant des données d'exécution d'autres objets.

Pour déclarer un objet de sorte qu'il ne soit pas exécuté au moment où l'agent utilisateur le lit, il faut placer l'attribut booléen `declare` dans l'élément `OBJECT` et identifier cette déclaration en affectant une valeur unique à son attribut `id`. Les instanciations ultérieures de l'objet se référeront à cet identifiant. Un élément `OBJECT` déclaré doit apparaître dans le document avant la première instance de cet élément `OBJECT`.

Un objet défini avec l'attribut `declare` est instancié chaque fois qu'un élément qui se rapporte à cet objet requiert une restitution de celui-ci

(par exemple, un lien qui se réfère à l'objet est activé, un autre objet qui s'y réfère est activé, etc.).

Les agents utilisateurs qui ne reconnaissent pas l'attribut `declare` doivent restituer le contenu de la déclaration de l'élément `OBJECT`.

L'élément `OBJECT` est aussi précieux que complexe. Pour en savoir plus, reportez-vous à la spécification HTML 4.01.

10.6. Résumé

- Un script est un programme écrit dans un langage de programmation particulier.
- Il existe deux types de script : les scripts côté serveur et les scripts côté client. Les scripts côté serveur sont situés et s'exécutent comme leur nom l'indique sur le serveur et peuvent envoyer un résultat vers la machine cliente. Les scripts côté client sont des programmes qui peuvent accompagner le document HTML ou bien y être directement incorporés. Ils s'exécutent sur la machine cliente (préservant ainsi les ressources du serveur).
- Un script à action unique est exécuté une seule fois, lors du chargement du document par l'agent utilisateur. Il apparaît dans le document à l'intérieur d'un élément `SCRIPT`.
- Un script répétitif est exécuté chaque fois que se produit un événement particulier.
- Vous insérez un script dans un document à l'aide de l'élément `SCRIPT`.
- HTML ne dépend pas d'un langage de script particulier. Les auteurs des documents indiquent explicitement aux agents utilisateurs le langage de chaque script. Vous pouvez le faire à l'aide d'une déclaration `META` placée dans l'élément `HEAD` ou en définissant l'attribut `type` de l'élément `SCRIPT`.
- L'élément `NOSCRIPT` permet de fournir un contenu de remplacement lorsque, pour une raison quelconque, un script n'est pas exécuté.
- Il est possible d'associer une action à un certain nombre d'événements intrinsèques qui se produisent lorsque l'utilisateur interagit avec l'agent utilisateur. Chacun des événements prend comme valeur un script.

- JavaScript est historiquement le premier langage de script pour le Web qui a permis aux développeurs Web d'exécuter de petits programmes sur le navigateur Web plutôt que sur le serveur Web. JScript est un langage de script très similaire développé par Microsoft.
- Le bon fonctionnement de JavaScript dépendait largement du navigateur employé. Un standard a été défini pour normaliser les langages de script de type JavaScript, ECMA-262 aussi appelé ECMAScript. C'est devenu un standard international (ECMA-262 et ISO-10262), ce qui en fait une des rares technologies Internet « non protocole » à avoir obtenu une normalisation d'un organisme officiel.
- VBScript est un langage de script propre à Microsoft, dérivé de leur langage de programmation Visual Basic.
- Un script CGI (*Common Gateway Interface*) est un script côté serveur. Ces scripts, souvent employés pour le traitement distant des formulaires, posent toutefois de nombreux problèmes.
- Un cookie est un petit fragment d'information qui reste stocké sur votre ordinateur après la clôture de la connexion. Les utilisateurs peuvent choisir d'accepter ou de refuser les cookies.
- HTML permet d'intégrer une applet dans votre page Web. Il ne s'agit toutefois pas obligatoirement d'une des célèbres applets Java : ce peut être une applet Python, Perl, etc. L'inclusion d'une applet s'effectuait autrefois à l'aide de l'élément `APPLET`, désormais déconseillé au profit de l'élément `OBJECT`.

Chapitre 11

Publication sur le Web

Identification du public 436
Accessibilité 438
Adaptation à plusieurs langues 456
Test du site 466
Publication sur le Web 472
Suivi de la fréquentation de votre site 478
Résumé 483

Au fil des chapitres précédents, vous avez découvert les principaux composants du langage HTML et avez construit un site Web minimaliste. Vous disposez toutefois désormais des connaissances nécessaires pour l'améliorer plus avant, grâce aux différentes techniques abordées.

Pour le moment, satisfait de votre première réalisation, vous souhaitez la placer sur le Web, pour qu'elle puisse y être consulté par tous. Par tous ? Vraiment ? Probablement pas. Tout site s'adresse à un public potentiel plus ou moins ciblé. Pour un site d'entreprise, dont le but est de promouvoir et de vendre un produit ou un service, la cible se compose des clients potentiels, ou du moins de ceux d'entre eux qui emploient Internet. Dès le début de la phase de conception de votre site, vous devez garder à l'esprit les caractéristiques de la cible visée et pour ce faire identifier votre public.

Ce public identifié, vous devrez probablement adapter votre site en fonction de ses demandes ou habitudes particulières.

Vous devrez ensuite améliorer l'accessibilité de votre site, pour tenir compte des personnes souffrant de divers handicaps, et éventuellement le rendre disponible en plusieurs langages. Enfin, vous le placerez sur le Net à l'aide d'un des multiples outils disponibles et le testerez à l'aide du plus grand nombre possible de navigateurs.

Ce chapitre s'intéresse à l'ensemble de ces démarches.

Éléments et attributs abordés dans ce chapitre :

```
IMG, alt
TABLE, summary, TH, TD, TR, headers, scope, abbr
NOFRAME
OBJECT, EMBEB, NOEMBED
lang, dir, BDO, dir, META, charset
```

11.1. Identification du public

L'identification des visiteurs probables d'un site Web est capitale. Nombreux sont les auteurs Web à commencer sans idée aucune de leur public potentiel. Ils se contentent d'assembler quelques idées, des éléments qui leur ont plu, généralement simplement pour apprendre à coder en HTML. Sans le savoir, leurs pages possèdent cependant un public : ceux qui partagent leurs thèmes d'intérêt (ceux qui figurent sur

les pages Web), ainsi que ceux qui possèdent une configuration matérielle et logicielle identique à celle du concepteur.

La composante fondamentale d'un site Web est l'information que vous tenez à partager. Le plus important est sans doute de savoir avec qui vous allez la partager. Ne surestimez pas le pouvoir d'un message ciblé. Internet est désormais trop vaste et trop diversifié pour garantir que vous serez capable d'atteindre tout le monde.

De toute évidence, le public le plus facile à cibler est celui d'un intranet d'entreprise. Le ou les navigateurs employés sont connus, ce qui simplifie grandement le codage. En revanche, de tels sites ne brillent généralement pas par l'innovation : ils sont essentiellement informatifs et destinés à aider les salariés à accomplir leur travail.

Si votre site s'adresse à des enfants, privilégiez les graphismes et les icônes significatives au texte. Les enfants préfèrent des images claires et colorées et sont souvent capables de comprendre le but d'une icône sans avoir à lire le texte.

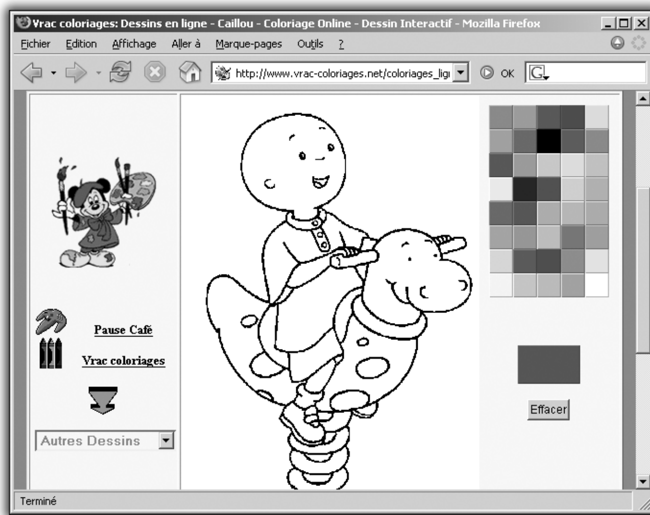


Figure 11.1 : *Un site destiné aux enfants insiste sur les icônes et les images*

En revanche, si le site concerne plutôt des personnes âgées (une population en forte croissance sur Internet), veillez à ce qu'il puisse s'adapter aux déficits visuels, ainsi qu'aux utilisateurs ayant du mal à utiliser le clavier ou la souris.



Figure 11.2 : *Le site de la FIDEV propose des versions adaptées aux personnes souffrant de déficits visuels*

Par contraste, si votre cible est constituée de *hard gamers*, ces fous de jeux vidéo, vous pouvez présumer sans grand risque d'erreur qu'ils disposent d'un écran de grande taille, d'une carte graphique performante, des derniers codecs, et probablement d'une liaison Internet à haut débit : dans un tel cas, pas question d'hésiter à abuser des images et vidéos !

Mais si vous suspectez que même un seul épileptique sensible à la lumière est susceptible de visiter votre site, oubliez les animations spectaculaires et les clignotements répétitifs. Vous devez en effet désormais tenir compte des personnes souffrant de handicaps et proposer des alternatives pour votre site. Aujourd'hui, de nombreux utilisateurs surfent sur le Web sans souris. Ils transmettent les ordres à l'agent utilisateur à l'aide du clavier ou de commandes vocales. La prise en compte de ces utilisateurs porte le nom d' « accessibilité du site ».

11.2. Accessibilité

L'ordinateur est un outil précieux, parfois indispensable à de nombreux utilisateurs qui souffrent de handicaps. Il procure l'accès à la communication et à l'enseignement indépendamment des capacités physiques. Internet a permis en outre la réalisation de véritables postes de travail virtuels. Qu'il s'agisse de recherche ou de courrier électronique, l'ordinateur aide les personnes handicapées à agir de façon interactive avec leur collègues, leurs amis et leur famille à un degré inimaginable dans le passé.

Il est de notoriété publique que la population des seniors croît régulièrement. Ceux-ci s'intéressent en outre de plus en plus à l'informatique, essentiellement grâce à Internet. Cette nouvelle vague

d’utilisateurs découvre toutefois que l’écran n’est pas toujours facile à lire et que la souris constitue un outil malcommode pour quelqu’un souffrant d’arthrite ou d’une quelconque affection du poignet. En tant que concepteur d’un site Web, vous ne devez ignorer aucune des parties de votre public potentiel.

Ayant déterminé qui va accéder à votre site, concevez-le de façon à faciliter son accès. Tenez compte de l’agent utilisateur probable, de la configuration matérielle envisageable, *etc*. Votre site doit-il être visité par des personnes âgées, des familles, le grand public ? Partez alors du principe qu’ils possèdent un écran de 17 pouces doté d’une résolution minimale de 800 x 600. Votre site est consacré à la programmation avancée ? Vérifiez qu’il gère bien le texte : de nombreux utilisateurs UNIX ne possèdent que des navigateurs en mode Texte seul. Votre site concerne les jeux vidéo ou s’adresse à des professionnels de l’imprimerie et du graphisme ? Vous pouvez sans crainte supposer qu’ils disposent d’un périphérique d’affichage de haut de gamme, probablement même supérieur au vôtre.

Voici les principaux pièges d’accessibilité rencontrés lors de la conception d’un site Web :

- Les cadres rendent la navigation difficile sans souris. De surcroît, ils posent problème sur de petits écrans à faible résolution et ne sont pas reconnus par les navigateurs textuels.
- Les tableaux sont assez mal reconnus par les navigateurs textuels, mais permettent d’afficher efficacement des données brutes. Si vous définissez la taille et la largeur, les utilisateurs possédant un affichage limité seront obligés de faire défiler la page pour lire l’intégralité du tableau. Un tableau peut souvent être remplacé par un autre élément structurel, comme un en-tête ou un pied de page, présent sur toutes les pages du site. Utilisez toujours le nouvel attribut `summary` pour procurer une alternative texte au tableau.
- Les images posent rarement de problèmes, même si de nombreux agents utilisateurs ne les affichent pas, par construction ou suite à un choix de l’utilisateur. Empêcher le chargement des graphismes permet de diminuer le temps de chargement d’une page Web ou facilite le travail d’un navigateur texte-parole. L’attribut `alt` placé dans l’élément `IMG` permet aux utilisateurs ayant désactivé les images d’en obtenir une description.
- Même si les graphismes animés permettent de capter l’attention, ils deviennent irritants lorsqu’ils apparaissent sur des pages de

contenu et non sur des pages de navigation. Une animation agressive peut rendre difficile la concentration d'utilisateurs sensibles à la lumière. Enfin, les utilisateurs peuvent bloquer les animations après leur chargement dans la plupart des navigateurs.

- Les formulaires ont subi de lourdes modifications en vue d'améliorer l'accessibilité. Le déplacement entre les champs d'un formulaire s'accomplit traditionnellement à l'aide de la touche ⇆ du clavier, parcourant ainsi les champs dans l'ordre où ils ont été rédigés en code HTML. Cela peut poser quelques problèmes, particulièrement s'il existe des liens hypertextes supplémentaires (comme des publicités) au milieu de la page. Utilisez les éléments `FIELDSET` et `LEGEND` ainsi que l'attribut `tabindex` (décrits dans le Chapitre 8) afin d'aider les utilisateurs à naviguer dans le formulaire sans la souris.



Cas de l'attribut `alt`

Vous devez toujours doter une image d'équivalent texte, mais la valeur de l'attribut `alt` doit satisfaire certains critères, faute de quoi il peut être considéré comme suspect. Respectez les règles suivantes :

Le texte ne doit pas dépasser 150 caractères (10 à 12 mots). N'écrivez pas :

```
<IMG src="images/couchersoleil.jpg"
alt="Coucher de soleil sur le lac de Linciel.
     Remarquez le jeu du vent sur les vagues
      et, à l'arri&egrave;re-plan, la vue sur la
      montagne du Destin">.
```

Le texte ne doit pas décrire une image qui sert de lien, mais la destination du lien.

```
<A href="images/couchersoleil.jpg">
  <IMG src="images/scouchersoleil.jpg"
       alt="Coucher de soleil sur le lac de Linciel">
Coucher de soleil</A>
```

Utilisez plutôt :

```
<A href="images/couchersoleil.jpg">
  <IMG src="images/scouchersoleil.jpg"
       alt="Lien vers l'image du coucher de soleil
           sur le lac de Linciel">Coucher de soleil</A>
```

Il ne doit pas contenir d'informations sur la taille du fichier, ni un nom de fichier ou une extension de fichier image. N'écrivez jamais :

```
<IMG src="images/couchersoleil.jpg"
    alt="Coucher de soleil sur le lac de Linciel.
   Nom : couchersoleil. Type : jpg. Taille : 64 500 octets">
```



Il ne doit pas servir de commentaire :

```
<IMG src="images/couchersoleil.jpg" alt="Ins&eacute;rer
l'image ici">
```

Enfin, les images d'espacement, les puces et les barres horizontales doivent également posséder un attribut `alt` défini.

Accessibilité des tableaux

Les tableaux sont une des structures HTML qui posent le plus de problèmes aux agents utilisateurs non visuels. Pourtant, HTML dispose désormais des outils nécessaires à en permettre une restitution parfaite. Il suffit de les mettre en œuvre.

Une cellule de tableau peut contenir deux types d'informations : des informations de rubrique (ou libellé) et des informations de données. Cette distinction permet aux agents utilisateurs de restituer les cellules de rubrique et de données différemment, même en l'absence de feuilles de style. Par exemple, les agents utilisateurs visuels peuvent présenter le texte des cellules de rubrique en caractères gras. Les synthétiseurs de parole peuvent restituer les informations de rubrique avec une inflexion de voix particulière.

L'élément `TH` définit une cellule qui contient une indication de rubrique. Les agents utilisateurs disposent de deux types d'informations de rubrique : le contenu de l'élément `TH` et la valeur de l'attribut facultatif `abbr`. Les agents utilisateurs doivent restituer soit le contenu de la cellule, soit la valeur de l'attribut `abbr`. Pour les médias visuels, cette dernière possibilité peut être adéquate quand il n'y a pas suffisamment de place pour restituer le contenu entier de la cellule. Pour les médias non visuels, l'attribut `abbr` peut s'employer comme abréviation des rubriques du tableau, quand celles-ci sont restituées en accompagnement du contenu des cellules concernées. Les attributs `headers` et `scope` permettent également aux auteurs d'assister les agents utilisateurs non visuels dans leur traitement des informations de rubrique.

L'élément `TD` définit une cellule qui contient des données. Une cellule peut être vide (ne pas contenir de données).

Les agents utilisateurs non visuels peuvent faire appel à certains attributs des éléments TD et TH pour restituer les cellules du tableau plus intuitivement :

- Pour une cellule de données particulière, l'attribut headers répertorie les cellules qui fournissent des indications de rubrique pertinentes. Chaque cellule de rubrique doit être nommée à l'aide de l'attribut id. Il n'est toutefois pas toujours possible de distinguer clairement cellules de rubrique et cellules de données. Servez-vous pour de telles cellules de l'élément TD accompagné d'un attribut id ou scope, selon que l'un ou l'autre est le plus adapté.
- Pour une cellule de rubrique donnée, l'attribut scope indique à l'agent utilisateur les cellules concernées par ces indications de rubrique. Les auteurs peuvent choisir d'employer cet attribut plutôt que l'attribut headers, s'ils considèrent qu'il convient mieux. Ces deux attributs remplissent la même fonction. L'attribut headers est en général nécessaire quand les rubriques se trouvent à une position inhabituelle par rapport aux données qui les concernent.
- L'attribut abbr spécifie un nom de rubrique abrégé pour les cellules de rubrique, pour que les agents utilisateurs puissent restituer plus rapidement les indications de rubrique.

Dans l'exemple suivant, nous assignons des indications de rubrique aux cellules en définissant l'attribut headers. Chaque cellule de la même colonne se rapporte à la même cellule de rubrique (*via* l'attribut id).

Listing 11-1 : Extrait du fichier headers.html

```
...
<TABLE border="1" summary="Ce tableau repr&eacute;sente
le nombre de livres emprunt&eacute;s par chaque membre de
la famille, et le nombre de livres non encore rendus.">
<CAPTION>Livres emprunt&eacute;s par chaque membre de
la famille</CAPTION>
<TR>
   <TH id="t1">Nom</TH>
   <TH id="t2">Livres</TH>
   <TH id="t3" abbr="Sortis">Non rendus ?</TH>
<TR>
   <TD headers="t1">Claude</TD>
   <TD headers="t2">10</TD>
   <TD headers="t3">2</TD>
<TR>
   <TD headers="t1">Martine</TD>
```

```
   <TD headers="t2">5</TD>
   <TD headers="t3">0</TD>
</TABLE>
...
```



Figure 11.3 : *Tableau employant l'attribut headers*

Un synthétiseur de parole pourrait restituer cette table comme suit :

```
Légende : Livres empruntés par chaque membre de la famille
Résumé : Ce tableau représente le nombre de livres
empruntés par chaque membre de la famille, et le nombre
de livres non encore rendus.
Nom : Claude, Livres : 10, Sortis : 2
Nom : Martine, Livres : 5, Sortis : 0
```

Remarquez la manière dont la rubrique *Non rendus ?* s'abrège en *Sortis*, grâce à l'attribut `abbr`.

Vous pourriez écrire le même exemple en substituant l'attribut `scope` à l'attribut `headers`.

Listing 11-2 : Extrait du fichier scope.html

```
...
<TABLE border="1" summary="Ce tableau repr&eacute;sente
le nombre de livres emprunt&eacute;s par chaque membre
de la famille, et le nombre de livres non encore rendus.">
<CAPTION>Livres emprunt&eacute;s par chaque membre
  de la famille</CAPTION>
<TR>
   <TH scope="col">Nom</TH>
   <TH scope="col">Livres</TH>
   <TH scope="col" abbr="Sortis">Non rendus ?</TH>
<TR>
   <TD>Claude</TD>
   <TD>10</TD>
   <TD>2</TD>
<TR>
   <TD>Martine</TD>
```

```
    <TD>5</TD>
    <TD>0</TD>
</TABLE>
...
```

Remarquez la valeur `col` de l'attribut `scope`, qui signifie « toutes les cellules de la colonne active ». Affiché dans un navigateur visuel, l'aspect est identique. Il en est de même avec un agent utilisateur non visuel.



Figure 11.4 :
Tableau employant l'attribut scope

Les autres valeurs de l'attribut `scope` permettent de construire des tableaux plus complexes :

Listing 11-3 : Extrait du fichier scope2.html

```
...
<TABLE border="1" cellpadding="5" cellspacing="2"
  summary="Quelques livres traduits par Fabrice
  Lemainque, class&eacute;s par titre, &eacute;diteur,
  r&eacute;sum&, code ISBN et prix.">
  <TR>
    <TH colspan="5" scope="colgroup">Traductions</TH>
  </TR>
  <TR>
    <TH scope="col" abbr="Titre">Titre du livre </TH>
    <TH scope="col" abbr="Ed.">&Eacute;diteur</TH>
    <TH scope="col">R&eacute;sum&eacute;</TH>
    <TH scope="col">Code ISBN</TH>
    <TH scope="col">Prix</TH>
  </TR>
  <TR>
    <TD scope="row">Wi-Foo, piratage et d&eacute;fense des réseaux
    ✂ sans fil</TD>
    <TD>CampusPress</TD>
    <TD>
Le d&eacute;veloppement des r&eacute;seaux sans fil
s'acc&eacute;l&egrave;re. Pourtant, les probl&egrave;mes
```

```
de s&eacute;curit&eacute; li&eacute;s &agrave; ce type de
r&eacute;seau sont largement sous-&eacute;stim&eacute;s. Ce
livre &eacute;tudie tant les m&eacute;thodes employ&eacute;es
pour p&eacute;n&eacute;trer un tel r&eacute;seau que celles
qui permettent de laisser les assaillants et les curieux
dehors. <EM>Niveau : Interm&eacute;diaire &agrave;
avanc&eacute;.</EM>
    </TD>
    <TD>2-7440-1948-8</TD>
    <TD>47 &euro;</TD>
  </TR>
  <TR>
    <TD scope="row">La Bible C++</TD>
    <TD>Micro Application</TD>
    <TD>
La r&eacute;f&eacute;rence absolue pour ma&icirc;triser
la programmation C++ !<BR>
Destin&eacute; aussi bien aux d&eacute;butants qu'aux
utilisateurs confirm&eacute;s, cet ouvrage,
r&eacute;f&eacute;rence incontest&eacute;e outre-atlantique
et traduit en Fran&ccedil;ais, est un v&eacute;ritable
outil de cours d&eacute;livrant aux utilisateurs des
bases solides et des connaissances pouss&eacute;es sur
le langage C++.  L'utilisateur dispose ici de toutes les
informations pour comprendre les fondements du langage,
résoudre ses probl&egrave;mes, puis d&eacute;velopper des
applications performantes avec le langage C++.
<EM>Niveau : Interm&eacute;diaire &agrave; avanc&eacute;.
</EM>
    </TD>
    <TD>2-7429-3717-X</TD>
    <TD>81 &euro;</TD>
  </TR>
</TABLE>
...
```

Figure 11.5 : *Tableau complexe employant l'attribut scope, avec des cellules fusionnées.*

Remarquez l'emploi de la valeur `row` pour l'attribut `scope`. Même si la première cellule de chaque ligne contient des données et non des indications de rubrique, l'attribut `scope` fait que la cellule de données se comporte comme une cellule de rubrique de ligne. Ceci permet aux synthétiseurs de parole de fournir à la demande le nom du livre concerné ou de le déclarer immédiatement avant le contenu de chaque cellule.

Il est parfois utile de catégoriser les cellules. Les utilisateurs qui parcourent une table avec un agent utilisateur basé sur la parole peuvent souhaiter entendre une explication sur le contenu d'une cellule, en plus du contenu en question. Pour ce faire, l'agent utilisateur pourrait lire les indications de rubrique associées avant le contenu de la cellule de données.

Les utilisateurs peuvent également vouloir des informations sur plusieurs cellules, auquel cas les indications de rubrique données au niveau de la cellule (attributs `headers`, `scope` et `abbr`) peuvent ne pas fournir le contexte adéquat. Examinez le tableau suivant, qui montre la répartition des dépenses au cours d'un voyage de plusieurs jours, à destination de deux villes, Toulouse et Montpellier.

"Catégorisation des cellules d'un tableau" - Mozilla Firefox

Catégorisation des cellules d'un tableau

Récapitulatif des dépenses de vacances

	Alimentation	Déplacement	Divers	Totaux
Toulouse				
17-07-05	37.74	32.00	15.00	
18-07-05	27.28	4.00	20.53	
Sous-totaux	65.02	36.00	35.53	136.52
Montpellier				
19-07-05	56.25	29.00	00.00	
20-07-05	35.00	0.00	36.00	
Sous-totaux	91.25	29.00	36.00	196.25
Totaux généraux	156.27	65.00	71.53	332.77

Figure 11.6 :
Dépenses de voyage

Le code de ce tableau est le suivant :

Listing 11-4 : Extrait du fichier vacances.html

```
...
<TABLE border="1" align="center"
```

```
          summary="Ce tableau r&eacute;sume les d&eacute;penses
          occasionn&eacute;s lors des vacances du mois de
          juillet &agrave; Toulouse et &agrave; Montpellier">
<CAPTION>
  R&eacute;capitulatif des d&eacute;penses de vacances
</CAPTION>
<TR>
  <TH></TH>
  <TH>Alimentation</TH>
  <TH>D&eacute;placement</TH>
  <TH>Divers</TH>
  <TH>Totaux</TH>
</TR>
<TR>
  <TH>Toulouse</TH>
  <TH></TH>
  <TH></TH>
  <TH></TH>
  <TD></TD>
</TR>
<TR align="right">
  <TD>17-07-05</TD>
  <TD>37.74</TD>
  <TD>32.00</TD>
  <TD>15.00</TD>
  <TD></TD>
</TR>
<TR align="right">
  <TD>18-07-05</TD>
  <TD>27.28</TD>
  <TD>4.00</TD>
  <TD>20.53</TD>
  <TD></TD>
</TR>
<TR align="right">
  <TD>Sous-totaux</TD>
  <TD>65.02</TD>
  <TD>36.00</TD>
  <TD>35.53</TD>
  <TD>136.52</TD>
</TR>
<TR>
  <TH>Montpellier</TH>
  <TH></TH>
  <TH></TH>
  <TH></TH>
  <TD></TD>
</TR>
<TR align="right">
  <TD>19-07-05</TD>
  <TD>56.25</TD>
  <TD>29.00</TD>
```

```
    <TD>00.00</TD>
    <TD></TD>
  </TR>
  <TR align="right">
    <TD>20-07-05</TD>
    <TD>35.00</TD>
    <TD>0.00</TD>
    <TD>36.00</TD>
    <TD></TD>
  </TR>
  <TR align="right">
    <TD>Sous-totaux</TD>
    <TD>91.25</TD>
    <TD>29.00</TD>
    <TD>36.00</TD>
    <TD>196.25</TD>
  </TR>
  <TR align="right">
    <TH>Totaux g&eacute;n&eacute;raux</TH>
    <TD>156.27</TD>
    <TD>65.00</TD>
    <TD>71.53</TD>
    <TD>332.77</TD>
  </TR>
  </TABLE>
  ...
```

L'utilisateur pourrait souhaiter extraire des informations du tableau sous la forme de requêtes comme :

- Quel a été le montant des frais de déplacement (essence, péage, etc.) ?
- Qu'ai-je dépensé en alimentation le 18 juillet ?
- Quel est le montant total des dépenses à Montpellier ?

Chaque requête implique un calcul par l'agent utilisateur, qui peut faire intervenir zéro ou plusieurs cellules. Pour déterminer, par exemple, les dépenses d'alimentation du 18 juillet, l'agent utilisateur doit reconnaître les cellules se rapportant à *Alimentation* et à *Dates* (particulièrement le 18 juillet), puis déterminer l'intersection de ces deux ensembles.

Pour satisfaire ce genre de requête, le modèle de table HTML 4 permet aux auteurs de classer les rubriques et les données des cellules par catégorie. Par exemple, pour le tableau des frais de déplacement, l'auteur pourrait regrouper les cellules de rubrique *Montpellier* et

Transports dans la catégorie *dépenses* et les quatre jours dans la catégorie *date*. Les trois questions précédentes se traduiraient alors comme suit :

- « Quel a été le montant des frais de déplacement ? » devient « Quelles sont toutes les cellules de données qui appartiennent à la catégorie *dépenses=Déplacement* ? »
- « Qu'ai-je dépensé en alimentation le 18 juillet ? » devient « Quelles sont toutes les cellules dans les catégories *dépenses=Alimentation* et *date=18-07-2005* ? »
- « Quel est le montant total des dépenses à Montpellier ? » devient « Quelles sont toutes les cellules dans les catégories *dépenses=Alimentation, Déplacement, Divers* et *ville=Montpellier* ? »

L'auteur catégorise une cellule de rubrique ou de données en spécifiant l'attribut `axis` de la cellule. Par exemple, dans le tableau des frais de déplacement, la cellule qui contient l'information *Montpellier* serait placée comme suit dans la catégorie *ville* :

```
<TH id="a6" axis="ville">Montpellier</TH>
```

Toute cellule qui contient des informations relatives à *Montpellier* devrait se rapporter à cette cellule de rubrique grâce à l'attribut `headers` ou à l'attribut `scope`. Ainsi, les frais d'alimentation du 18-07-2005 devraient être balisés pour se rapporter à l'attribut `id` (dont la valeur ici est `a6`) de la cellule de rubrique :

```
<TD headers="a6">27.28</TD>
```

Chaque attribut `headers` fournit une liste de références `id`. Vous pouvez donc catégoriser une cellule donnée par divers moyens (ou selon un nombre quelconque de rubriques).

Voici le balisage du tableau des dépenses avec des indications de catégorie. Les modifications apportées au fichier précédent figurent en gras :

Listing 11-5 : Extrait du fichier vacances2.html.

```
...
</CAPTION>
<TR>
  <TH></TH>
  <TH id="a2" axis="dépenses">Alimentation</TH>
  <TH id="a3" axis="dépenses">D&eacute;placement</TH>
  <TH id="a4" axis="dépenses">Divers</TH>
  <TD>Sous-totaux</TD>
```

```
</TR>
<TR>
  <TH id="a6" axis="ville">Toulouse</TH>
  <TH></TH>
  <TH></TH>
  <TH></TH>
  <TD></TD>
</TR>
<TR align="right">
  <TD id="a7" axis="date">17-07-05</TD>
  <TD headers="a6 a7 a2">37.74</TD>
  <TD headers="a6 a7 a3">32.00</TD>
  <TD headers="a6 a7 a4">15.00</TD>
  <TD></TD>
</TR>
<TR align="right">
  <TD id="a8" axis="date">18-07-05</TD>
  <TD headers="a6 a8 a2">27.28</TD>
  <TD headers="a6 a8 a3">4.00</TD>
  <TD headers="a6 a8 a4">20.53</TD>
  <TD></TD>
</TR>
<TR align="right">
  <TD>Sous-totaux</TD>
  <TD>65.02</TD>
  <TD>36.00</TD>
  <TD>35.53</TD>
  <TD>136.52</TD>
</TR>
<TR>
  <TH id="a10" axis="ville">Montpellier</TH>
  <TH></TH>
  <TH></TH>
  <TH></TH>
  <TD></TD>
</TR>
<TR align="right">
  <TD id="a11" axis="date">19-07-05</TD>
  <TD headers="a10 a11 a2">56.25</TD>
  <TD headers="a10 a11 a3">29.00</TD>
  <TD headers="a10 a11 a4">00.00</TD>
  <TD></TD>
</TR>
<TR align="right">
  <TD id="a12" axis="date">20-07-05</TD>
  <TD headers="a10 a12 a2">35.00</TD>
  <TD headers="a10 a12 a3">0.00</TD>
  <TD headers="a10 a12 a4">36.00</TD>
  <TD></TD>
</TR>
...
```

L'aspect du tableau dans un navigateur visuel est strictement identique.

"Catégorisation des cellules d'un tableau" - Mozilla Firefox

Fichier Edition Affichage Aller à Marque-pages Outils ?

Catégorisation des cellules d'un tableau

Récapitulatif des dépenses de vacances

	Alimentation	Déplacement	Divers	Totaux
Toulouse				
17-07-05	37.74	32.00	15.00	
18-07-05	27.28	4.00	20.53	
Sous-totaux	65.02	36.00	35.53	136.52
Montpellier				
19-07-05	56.25	29.00	00.00	
20-07-05	35.00	0.00	36.00	
Sous-totaux	91.25	29.00	36.00	196.25
Totaux généraux	156.27	65.00	71.53	332.77

Terminé

Figure 11.7 :
Tableau des dépenses de voyage avec catégorisation des cellules

Pourtant, ce balisage permet aux agents utilisateurs d'éviter de troubler l'utilisateur avec des informations parasites. Par exemple, si un synthétiseur de parole devait dire tous les nombres figurant dans la colonne *Alimentation* de cette table, en réponse à la question « Quelle est la totalité de mes frais de repas ? », l'utilisateur serait incapable de distinguer entre les dépenses du jour et les sous-totaux ou les totaux. En catégorisant soigneusement les cellules de données, les auteurs permettent aux agents utilisateurs de faire des distinctions sémantiques importantes lors de la restitution.

Il n'existe pas de limite quant à la manière dont les auteurs peuvent catégoriser les informations contenues dans une table. Dans cette table des frais de déplacement, par exemple, on pourrait rajouter les catégories *sous-totaux* et *totaux*.

Les agents utilisateurs, notamment les synthétiseurs de parole, peuvent combiner les informations communes à plusieurs cellules résultant d'une requête. Par exemple, si l'utilisateur demande « Qu'ai-je dépensé en alimentation à Montpellier ? », l'agent utilisateur déterminerait d'abord les cellules en question (19-07-05 : 56,25 ; 20-07-05 : 35,00) puis

restituerait ces informations. Un agent utilisateur disant ces informations pourrait les lire ainsi :

```
Ville : Montpellier. Date : 19-07-05. Dépenses,
Alimentation : 56,25

Ville : Montpellier. Date : 20-07-05. Dépenses,
Alimentation : 35,00
```

ou, de manière plus concise :

```
Montpellier, 19-07-05, Alimentation : 56,25

Montpellier, 20-07-05, Alimentation : 35,00
```

Une restitution encore plus économe combinerait les informations communes et les réorganiserait :

```
Montpellier, Alimentation, 19-07-05 : 56,25

                           20-07-05 : 35,00
```



L'attribut `axis`

La spécification HTML 4.01 n'oblige pas les agents utilisateurs à gérer les informations fournies par l'attribut `axis`. Elle n'émet pas non plus de recommandation sur la manière dont les agents utilisateurs peuvent présenter les informations de l'attribut `axis` aux utilisateurs, ni sur la façon dont les utilisateurs peuvent effectuer des requêtes vers l'agent utilisateur.

En l'absence d'indications de rubrique fournies par l'attribut `scope` ou l'attribut `headers`, les agents utilisateurs peuvent néanmoins construire des informations de rubrique en fonction de l'algorithme suivant, dont l'objectif consiste à rechercher une liste ordonnée de rubriques.

- À partir de la position de la cellule, rechercher d'abord des cellules de rubrique de ligne vers la gauche. Puis rechercher des cellules de rubrique de colonne vers le haut. La recherche dans une direction s'achève quand le bord du tableau est atteint ou bien quand on trouve une cellule de données qui fait suite à une cellule de rubrique.

- Les rubriques de ligne sont insérées dans la liste, dans l'ordre où elles apparaissent dans la table. Pour les tableaux de gauche à droite, les rubriques s'insèrent de gauche à droite.
- Les rubriques de colonne sont insérées après les rubriques de ligne, dans l'ordre où elles apparaissent dans le tableau, de haut en bas.
- Si une cellule de rubrique possède un attribut `headers`, les rubriques référencées par cet attribut sont insérées dans la liste et la recherche s'achève pour la direction courante.
- Les cellules `TD` qui possèdent un attribut `axis` sont également traitées comme des cellules de rubrique.

Accessibilité des jeux d'encadrement

Les jeux d'encadrement font également partie des structures délicates à interpréter par les agents utilisateurs non visuels. Les remèdes sont toutefois ici presque inexistants. La seule solution, comme cela était évoqué au Chapitre 7, consiste à employer un élément `NOFRAMES` et à fournir un lien alternatif vers une version dépourvue de cadres.

```
<NOFRAMES>
<H1>Bienvenue chez ORIGNAL TRADUCTIONS !</H1>
Votre agent utilisateur ne reconna&icirc;t malheureusement pas les
jeux d'encadrement. Cliquez <A href="oritexte.html">ici</A> pour
acc&eacute;der au site alternatif en version purement textuelle.
</NOFRAMES>
```

Avec l'agent utilisateur pwWebspeak, (n'oubliez pas que pwWebspeak « lit » ce qui est affiché), vous obtenez l'aspect visuel qui est présenté dans la figure suivante.

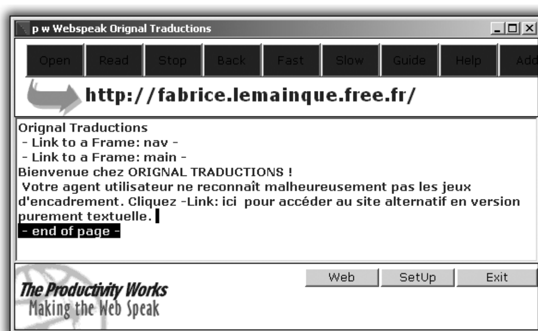


Figure 11.8 : *pwWebSpeak ne reconnaît pas les jeux d'encadrement*

Le navigateur en texte seul Linx affiche ce qui est présenté figure suivante.

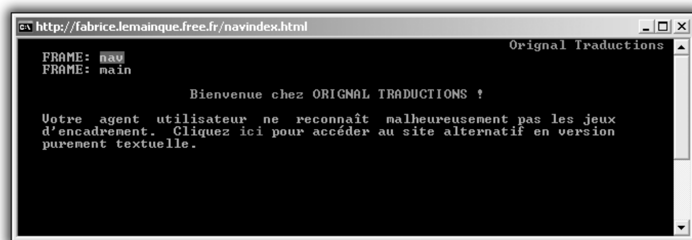


Figure 11.9 : *Linx ne reconnaît pas non plus les jeux d'encadrement*

Enfin, un navigateur visuel ne reconnaissant pas les jeux d'encadrement pourrait afficher ce qui est présenté dans la figure qui suit.



Figure 11.10 : *Affichage avec un navigateur visuel ne reconnaissant pas les jeux d'encadrement*

En l'absence de l'élément NOFRAME, les agents utilisateurs afficheraient une page vide !

Modules complémentaires (plug-in)

Pourquoi aborder ce sujet dans une section traitant de l'accessibilité ? Parce que, une fois encore, le choix d'imposer le chargement et l'installation de modules complémentaires (ou *plug-in*) pour accéder à un site dépend du public visé.

Si celui-ci est composé d'internautes avertis, il est probable qu'ils disposent déjà des principaux modules complémentaires et ne rechigneront pas à les installer éventuellement si ce n'est pas le cas. En revanche, de nombreux utilisateurs moins bien informés ne se donneront

pas la peine de télécharger un programme complémentaire pour visionner certaines sections de votre site. Prévoyez toujours une solution alternative pour ce type de visiteurs !

Voici quelques exemples de contenus susceptibles de réclamer un module complémentaire :

- Texte au format PDF (*Portable Document format*) : Adobe Acrobate Reader.
- Musique : RealAudio, MIDI, WAV, QuickTime.
- Vidéo : MPEG, AVI.
- Animations : Flash/Shockwave (Macromedia), applets Java.
- Mondes virtuels : VRML .

Adobe Acrobat Reader est un célèbre *plug-in* qui permet de lire et d'imprimer des documents du Web ayant été formatés pour l'impression. Les utilisateurs peuvent avoir accès aux formulaires gouvernementaux, aux brochures collectives et à bien d'autres informations. Si l'apparence de l'information que vous voulez présenter est très importante, envisagez de la présenter comme un fichier Adobe Acrobat.

Shockwave est également très répandu. Proposé par Macromédia, il permet aux utilisateurs de voir des animations sophistiquées ainsi que des films créés avec le logiciel Director de Macromedia.

Les applets Java nécessitent l'installation d'une machine Java virtuelle (JVM, *Java Virtual Machine*), toutefois parfois intégrée au navigateur (cas d'Internet Explorer).

L'incorporation de ce type d'éléments dans une page Web s'effectue à l'aide de l'élément `OBJECT` déjà abordé. L'élément `OBJECT` a pour but de gérer à terme tous les contenus non textuels. Par exemple, des images simples, des films et des applets Java peuvent tous être affichés à l'aide de l'élément `OBJECT`.

Cet élément permet de créer différentes possibilités pour les utilisateurs qui ne peuvent visualiser vos fichiers multimédias : un utilisateur peut voir du texte, un autre une image fixe, un dernier une vidéo de haute qualité.

Comme le montre le tableau suivant, certains modules complémentaires sont toutefois largement installés :

Tableau 11.1 : *Les technologies (plug-in) installées sur les postes des internautes en février 2008. (Source : http://www.web-stats.org/k/pluggins-par-pluggins.html)*

Rang	Produit	Présence
1	Java	98,7 %
2	Javascript	98,5 %
3	Flash	96,9 %
4	QuickTime	49,9 %
5	Real Player	27,6 %
6	Shockwave	24,9 %
7	PDF	13,3 %

11.3. Adaptation à plusieurs langues

Vous pourriez vouloir proposer des versions de votre site adaptées à plusieurs cultures et langues, afin d'élargir votre public. Certains parlent pour ce faire d'*internationalisation* du site, terme souvent confondu avec celui de *localisation*.

Pour simplifier, l'internationalisation est le processus par lequel un produit est rendu disponible aux utilisateurs d'autres langues et d'autres cultures. Pour HTML 4.0, il s'agit essentiellement de prendre en charge les langues qui se lisent de droite à gauche, comme l'hébreu et l'arabe, ainsi que celles fondées sur des idéogrammes, comme le japonais et le chinois. En revanche, la localisation consiste à adapter un produit ou une technologie à un marché local composé de personnes employant une langue spécifique. Autrement dit, vous internationalisez votre site en localisant différentes versions de vos pages.

L'ensemble de caractères Unicode désormais adopté par HTML est la réponse ultime au problème des différentes langues. Il dispose de plus de 25 000 caractères différents et est destiné à représenter les langues du monde entier (plus quelques autres, comme le klingon - attribut `lang="x-klingon"`, `x` - spécifiant une langue expérimentale). Vous

trouverez des informations complémentaires sur Unicode à l'adresse **www.unicode.org/**

Vous spécifiez la langue d'un élément HTML à l'aide de son attribut `lang`.

Attribut lang

L'attribut `lang` spécifie la langue de base dans laquelle sont écrits les valeurs des attributs et le contenu textuel d'un élément. La valeur par défaut de cet attribut n'est pas définie. Cet attribut s'applique ou non à un attribut selon sa syntaxe et sa sémantique et l'opération en jeu.

Vous devez être capable d'écrire dans la langue humaine de votre document. Le Web ne se comporte pas comme un traducteur universel, capable par exemple de modifier une page Web du français au japonais. Vous devez le traduire vous-même en japonais, dans le pire des cas à l'aide d'un des nombreux programmes de traduction disponibles sur le Web.

L'indication de langue procurée par l'attribut `lang` peut permettre à un agent utilisateur de contrôler la restitution de diverses manières, en respectant les pratiques culturelles en usage dans une langue donnée.

Si le but de l'attribut `lang` est de permettre aux agents utilisateurs de représenter un contenu de façon plus pertinente, il n'incite aucunement à représenter de façon moins pertinente les caractères atypiques pour une langue particulière. Un agent utilisateur doit toujours faire de son mieux pour représenter tous les caractères, quelle que soit la valeur spécifiée par `lang`.

Par exemple, imaginez que vous ayez placé des caractères de l'alphabet grec au milieu d'un texte en français :

```
<P><Q lang="fr">Ses super-pouvoirs étaient la conséquence d'un
✂ rayonnement &gamma;</Q>, expliqua-t-il.</P>
```

L'agent utilisateur doit s'efforcer de représenter le contenu français de manière appropriée (dans sa façon de gérer les guillemets par exemple) et faire son possible pour représenter le caractère γ, même si ce n'est pas un caractère français.

Remarquez que Firefox

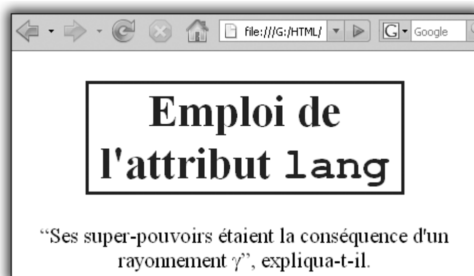


Figure 11.11 : *Aperçu du code précédent avec le navigateur Firefox*

et Internet Explorer



Figure 11.12 : *Aperçu du code précédent avec le navigateur Internet Explorer*

n'interprètent pas de façon identique les guillemets normalement induits par l'élément `Q`, comme nous l'avons vu lors de l'étude de cet élément.

L'attribut `lang` reçoit comme valeur un code qui identifie une langue parlée, écrite, ou utilisée d'une manière ou d'une autre pour la communication d'informations entre personnes. Les langages informatiques sont explicitement exclus des codes de langue. Le document RFC1766 définit et explique les codes de langue qui doivent être utilisés dans les documents HTML.

Brièvement, les codes de langue consistent en un code principal et une suite éventuellement vide de sous-codes :

```
code-de-langue = code-principal ("-" sous-code)*
```

Le tableau suivant présente certains codes de langue.

Valeur de `lang`	Langage
`ar`	Arabe
`de`	Allemand
`el`	Grec
`en`	Anglais
`en-cockney`	Version est-londonienne de l'anglais
`es`	Espagnol
`fr`	Français
`fr-CA`	Français canadien
`he`	Hébreu
`hi`	Hindi
`i-navajo`	La langue navajo parlée par certains Indiens d'Amérique
`it`	Italien
`ja`	Japonais
`nl`	Néerlandais
`pt`	Portugais
`ru`	Russe
`sa`	Sanscrit
`ur`	Ourdou
`x-klingon`	Klingon (pensez à StarTrek). Le code principal `x` signale une langue expérimentale.
`zh`	Chinois

***Tableau 11.2 :** Valeurs de l'attribut* `lang`

Un élément hérite de l'indication du code de langue selon l'ordre de priorité suivant (de la priorité la plus élevée à la plus faible) :

- L'attribut `lang` spécifié pour l'élément lui-même.
- L'élément parent le plus proche, dont l'attribut `lang` est spécifié.
- L'en-tête HTTP `Content-Language` (qui peut être configuré au sein d'un serveur). Par exemple : `Content-Language: fr-CA`.

- Les valeurs par défaut de l'agent utilisateur et les préférences de l'utilisateur.

Dans l'exemple suivant, la langue principale du document est le français (fr). Un paragraphe est déclaré comme étant en espagnol (es), après lequel la langue principale redevient le français. Le paragraphe qui vient ensuite inclut une phrase enchâssée en japonais (ja), après laquelle la langue principale redevient le français.

Listing 11-6 : Extrait du fichier depression.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML lang="fr">
<HEAD>
<TITLE>D&eacute;pression multilingue</TITLE>
...
</HEAD>
<BODY>
...A propos de la d&eacute;pression...
<P lang="es">¿Tiene Usted Depresi&oacute;n?
<P> La d&acute;pression au Japon, connue sous le nom de <EM
lang="ja">Utsubyo</EM>, y est très fr&eacute;quente :
probablement une personne sur sept en souffre...
</BODY>
</HTML>'
```

La figure suivante présente l'aspect de cette page dans un navigateur.



Figure 11.13 : *Texte multilingue*



Héritage

Les cellules d'un tableau peuvent hériter des valeurs de `lang` non de leur parent mais de la première cellule d'un groupe.

Lorsqu'un agent utilisateur ajuste la restitution en fonction des indications de langue (en comparant les codes de langue de la feuille de style et les valeurs de l'attribut `lang`), il doit toujours donner la préférence aux codes qui coïncident parfaitement. Il peut toutefois considérer la correspondance des codes principaux comme suffisante. Ainsi, si l'attribut `lang` est spécifié pour l'élément HTML avec la valeur `fr-CA`, l'agent utilisateur devrait-il préférer les indications de style qui correspondent d'abord à `fr-CA`, puis à la valeur plus générale `fr`.

Les agents utilisateurs ne doivent pas utiliser l'attribut `lang` pour déterminer la direction du texte.

Attribut dir

Outre spécifier la langue d'un document à l'aide de l'attribut `lang`, il peut être nécessaire de définir la directionnalité de base (de gauche à droite ou de droite à gauche) de certaines parties du texte d'un document, de la structure d'un tableau, etc. Vous y parvenez à l'aide de l'attribut `dir`, qui spécifie la direction de base d'un texte neutre par rapport à la direction (autrement dit, un texte dépourvu de directionnalité inhérente telle que définie dans la norme Unicode) dans le contenu ou les valeurs d'attribut d'un élément. Il spécifie aussi la directionnalité des tableaux. Les valeurs possibles sont :

- `LTR` (*left to right*) : texte ou tableau allant de gauche à droite.
- `RTL` (*right to left*) : texte ou tableau allant de droite à gauche.

La spécification Unicode attribue une directionnalité aux caractères et définit un algorithme complexe pour déterminer la directionnalité correcte d'un texte. Si un document ne contient pas de caractères affichables droite à gauche, l'agent utilisateur conforme n'est pas tenu d'appliquer l'algorithme bidirectionnel Unicode. Si un document contient des caractères droite à gauche, et si l'agent utilisateur affiche ces caractères, l'agent utilisateur doit utiliser l'algorithme bidirectionnel.

Même si la norme Unicode spécifie des caractères spéciaux concernant la direction du texte, HTML offre des structures de balisage de niveau supérieur qui accomplissent la même chose : l'attribut `dir` (à ne pas confondre avec l'élément `DIR`) et l'élément `BDO`. Ainsi, pour faire une citation en hébreu, il est plus intuitif d'écrire :

```
<Q lang="he" dir="rtl">...une citation en hébreu...</Q>
```

que de faire appel à des caractères Unicode :

```
&#x202B;&#x05F4;...une citation en hébreu...&#x05F4;&#x202C;
```



Figure 11.14 :
Attribut dir avec Firefox



Figure 11.15 :
Attribut dir avec Internet Explorer

Comme le montrent les copies d'écran précédentes, ces deux lignes de code ont un effet identique… mais pas celui attendu, puisque le texte est bien écrit de gauche à droite ! Ni Firefox ni Internet Explorer ne semblent reconnaître pour le moment l'attribut `dir`.

En réalité, ces deux navigateurs semblent interpréter l'attribut `dir` comme similaire à un attribut `align`. Ainsi, le code suivant :

```
<HTML dir="RTL">
<HEAD>
<TITLE>...un titre allant normalement de droite à gauche...</TITLE>
</HEAD>
<BODY>
```

```
<H1>Titre allant aussi de droite à gauche...</H1>
<P dir="ltr">...texte allant de gauche à droite...</P>
<P>...encore un texte de droite à gauche...</P>
</BODY>
</HTML>
```

donne, avec ces deux navigateurs, un résultat strictement équivalent à :

```
<HTML>
<HEAD>
<TITLE>...un titre allant normalement de droite à gauche...</TITLE>
</HEAD>
<BODY>
<H1 align="right">Titre allant aussi de droite à gauche...</H1>
<P align="left">...texte allant de gauche à droite...</P>
<P align="right">...encore un texte de droite à gauche...</P>
</BODY>
</HTML>
```

Les figures suivantes présentent l'aspect de ces deux codes avec Firefox.



Figure 11.16 :
Attribut dir avec Firefox



Figure 11.17 :
Attribut align avec Firefox

Heureusement, il est possible de désactiver l'algorithme bidirectionnel Unicode et de spécifier une directionnalité à l'aide de l'élément `BDO` (*BiDirectional Override*).

Élément BDO

Si l'algorithme bidirectionnel et l'attribut `dir` devraient suffire à gérer les changements de direction enchâssés, il semble toutefois qu'ils ne

soient pas parfaitement pris en charge pour le moment. L'élément BDO, dont les balises d'ouverture et de fermeture sont obligatoires, permet de définir une directionnalité indépendamment de l'algorithme bidirectionnel sur des passages choisis du texte.

Cet élément possède un attribut obligatoire dir, identique à celui vu ci-dessus, mais qui prévaut sur la directionnalité inhérente des caractères telle que définie dans Unicode.

Reprenons l'exemple précédent, cette fois avec des éléments BDO :

```
<HTML>
<HEAD>
<TITLE><BDO dir="rtl">Eléments BDO (droite à gauche)</BDO></TITLE>
</HEAD>
<BODY>
<H1><BDO dir="rtl">Titre allant aussi de droite à gauche...</BDO></H1>
<P><BDO dir="ltr">...texte allant de gauche à droite...</BDO></P>
<P><BDO dir="rtl">...encore un texte de droite à gauche...</BDO></P>
</BODY>
</HTML>
```

Comme le montre la figure suivante, l'aspect obtenu est celui souhaité, si ce n'est que l'élément BDO n'est pris en compte que dans le corps du document, et non dans un élément TITLE.

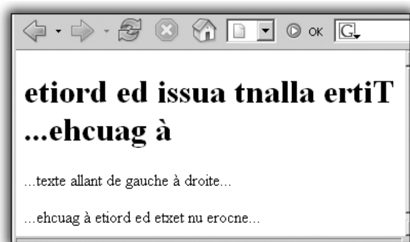


Figure 11.18 : *Éléments BDO et attributs dir avec Firefox*

Attention toutefois, seuls les navigateurs les plus récents reconnaissent l'élément BDO.

D'une façon générale, l'algorithme bidirectionnel Unicode est d'une grande complexité et intervient dans de nombreux domaines. Si cela vous intéresse, reportez-vous à la spécification HTML 4.01 pour en apprendre plus à son sujet.

Enfin, prenez garde à tester votre page en langue étrangère à l'aide d'un navigateur reconnaissant cette langue. Si l'aspect n'est pas correct, revenez en arrière et placez un attribut lang dans tout élément relatif au

contenu de la page. Il sera parfois nécessaire d'avoir recours à un élément `<BASEFONT>` dans la section `<HEAD>`, afin d'imposer l'utilisation d'une police exotique.

Langues et jeux de caractères

Si vous réalisez une page dans une langue autre qu'européenne, n'oubliez pas de modifier la page de caractères du document à l'aide de l'attribut `charset`. Certains alphabets, tel l'alphabet cyrillique, possèdent des caractères difficilement affichables à l'aide des caractères d'Europe de l'Ouest Latin-1 (Western European), nommés ISO-8859-1. Les caractères cyrilliques seront trouvés dans la page *Windows-1251*, tandis que *SHIFT_JIS* et *EUC-JP* produisent les caractères japonais. Vous pouvez spécifier le jeu de caractères pour un document en cyrillique comme ceci :

```
<META http-equiv="Content-Type" content="text/html; charset=
✂ Windows-1251">
```

L'image suivante montre une page russe, qui présente des caractères cyrilliques,

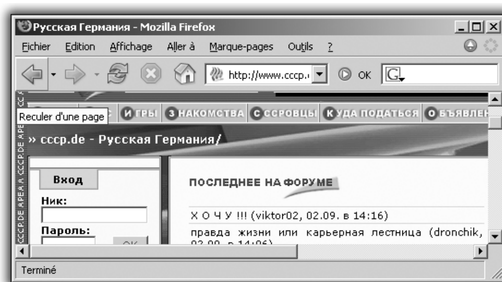


Figure 11.19 :
Site russe, à l'adresse http://www.cccp.de/

Voici le code source de cette page avec la spécification du jeu de caractères pour cette page.

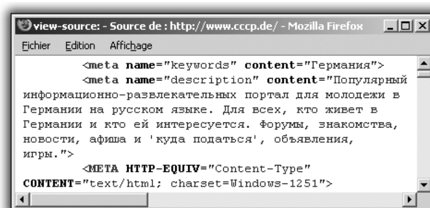


Figure 11.20 :
Code source de cette page, montrant le choix du jeu de caractères Windows-1251.

Avec de nombreux navigateurs, vous pouvez forcer le choix du type d'encodage de la page. Par exemple, avec Firefox, vous choisissez **Affichage / Encodage des caractères**.



Figure 11.21 :
Modification de l'encodage de la page avec Firefox

Choisir pour la page russe le jeu occidental n'est toutefois pas sage : comme le montre la figure suivante, la page devient incompréhensible.

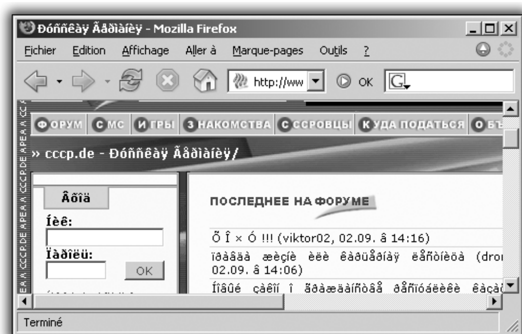


Figure 11.22 :
Aspect du site précédent, en imposant un encodage occidental ISO-8859-1.

11.4. Test du site

Nous avons déjà insisté sur le fait qu'il fallait toujours tester un site Web avec autant de navigateurs que possible. La plupart d'entre eux sont disponibles gratuitement, même s'il ne s'agit que de versions de démonstration. D'une façon générale, comme cela a été évoqué dans le Chapitre 2, il est préférable de tester le site au moins sous Internet Explorer 6.0 et 7.0 et sous Firefox 1 et 2.

Dans le cas d'un public qui comprend des utilisateurs à difficultés visuelles, le mieux est un navigateur texte-parole. Ceux-ci sont plutôt onéreux, mais une version de démonstration de pwWebSpeack est disponible à l'adresse **www.prodworks.com** Une alternative consiste à utiliser un navigateur en texte seul, comme Lynx (**http://lynx.browser.org/**), afin de visualiser votre site. Prenez toujours la précaution d'examiner votre site sous un navigateur graphique après avoir désactivé les graphismes. Voici comment désactiver les graphismes avec les deux principaux navigateurs :

- Avec Internet Explorer, choisissez **Outils / Options Internet**. Cliquez sur l'onglet **Avancé**, puis descendez dans la liste et décochez dans la section **Multimédia** l'option *Afficher les images*.

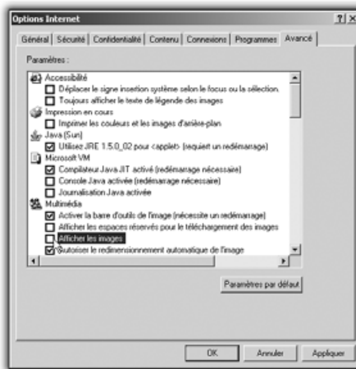


Figure 11.23 :
Désactivation des images avec Internet Explorer

- Avec Firefox, choisissez **Outils / Options**. Choisissez **Fonctionnalités Web** dans le volet de gauche, puis décochez dans le volet de droite l'option *Charger les images* .

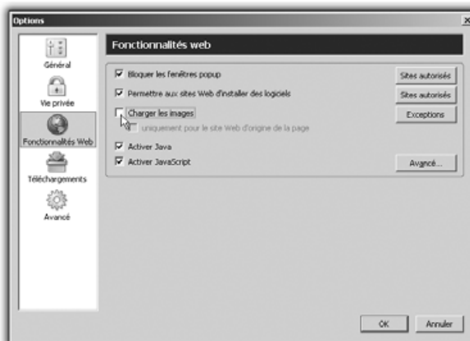


Figure 11.24 :
Désactivation de graphismes avec Firefox

Remarquez qu'un navigateur à images désactivées (ou en texte seul) supprime toutes les indications des feuilles de style. Tout graphisme éventuellement présent sur la page n'affiche que le texte présent dans son attribut `alt`.

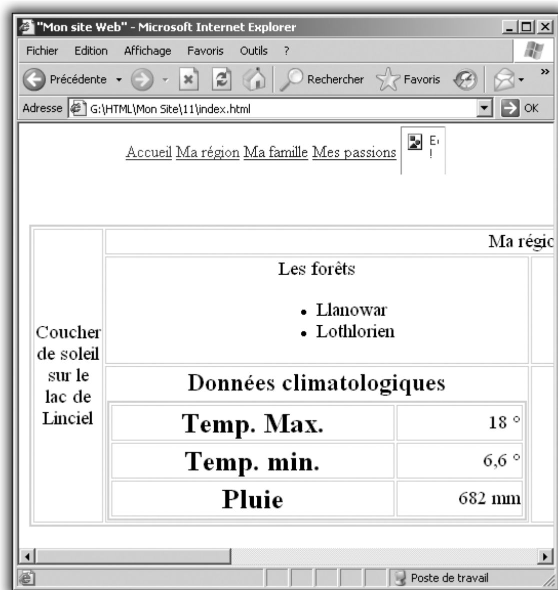


Figure 11.25 : *Page Ma région dans Internet Explorer, les graphismes étant désactivés.*

Bien évidemment, si vous réalisez une page pour un intranet, testez-le avec le navigateur utilisé par l'entreprise, pourvu de tous les programmes complémentaires (plug-in) installés de façon standard.

Prenez garde aux problèmes de compatibilité entre navigateurs. La page est-elle parfaite avec un autre navigateur ? Si vous avez conçu votre site en pensant à un navigateur unique, sans pouvoir exercer de contrôle sur le navigateur réellement utilisé, n'imaginez pas que votre page sera « parfaite » pour chaque visiteur. Examinez votre page avec au moins un autre navigateur, et observez ses transformations, en vérifiant qu'elle ne perd pas sa signification. Si tel était le cas, essayez de la modifier afin que le message perçu reste identique quels que soient le navigateur et la configuration matérielle de l'utilisateur.

Il est bien sûr difficile de disposer de tous les navigateurs sur son ordinateur. Plusieurs solutions sont possibles :

- **Partage du test des ressources Web avec d'autres**. Il peut s'agir d'amis ou de relations, qui possèdent une architecture matérielle et/ou un navigateur différents des vôtres. Vous pouvez également solliciter, sur certains groupes de discussion, l'examen de votre site. Certains développeurs peuvent pousser le professionnalisme jusqu'à vous envoyer une capture d'écran ! En attirant l'attention des testeurs sur certains problèmes spécifiques que vous suspectez, les réponses reçues devraient vous permettre de résoudre tout problème potentiel.
- **Emploi de sites de test disponibles sur le Net** (comme, par exemple, AnyBrowser.com). Vous pouvez :
  - tester votre site à l'aide d'une simulation de navigateur de troisième génération ;

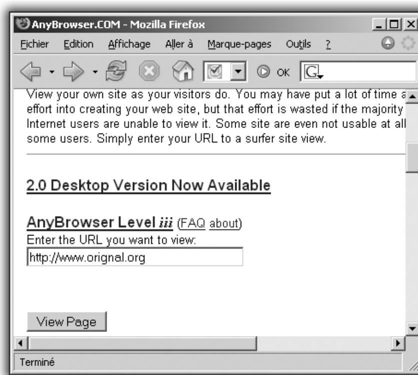


Figure 11.26 : *Test d'un site avec un navigateur de troisième génération*

  - spécifier plus précisément un niveau de compatibilité avec les spécifications HTML ;
  - tester l'affichage sur une WebTV.

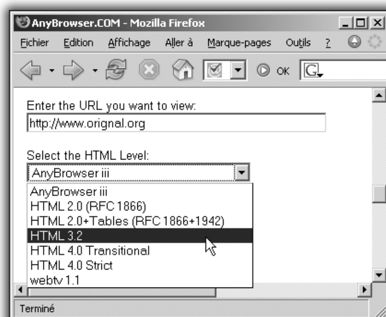


Figure 11.27 : *Test d'un site en choisissant le niveau de compatibilité du navigateur*

Pensez également à modifier votre résolution et à faire varier la taille de la fenêtre du navigateur. Des surprises ne manqueront pas de s'ensuivre…

Ne poussez toutefois pas la précision à l'extrême : les différences entre navigateurs sont inévitables. Tant que le contenu reste lisible, ne modifiez pas votre page parce qu'un élément diffère légèrement d'un autre.

Tout au long de la phase de test, vous devrez répondre à plusieurs questions. Tout d'abord, celles qui concernent directement le contenu :

- Les informations détaillées sur le contenu sont-elles directement ou rapidement disponibles ?
- Tout document doit posséder un titre. Vos titres résument-ils bien le contenu des pages ?
- L'esthétique des pages est-elle de bon goût ? Originale ? Attrayante ?
- N'y a-t-il rien qui puisse choquer inutilement un lecteur d'une culture différente de la vôtre ?
- Le raffinement de votre page d'accueil n'est-il pas excessif au point de pénaliser lourdement le temps de chargement (image trop grande ou texte trop long) ? La page d'accueil signale-t-elle la configuration optimale pour visionner votre site ?
- Tous les liens sont-ils bien définis ?
- Toutes les images sont-elles bien présentes ? Comptent-elles toutes bien un texte alternatif ? Avez-vous pensé à limiter leur taille ?

Vous pouvez en outre utilement vérifier le bon balisage de votre document, soit à l'aide du service proposé par le W3C à l'adresse **http://validator.w3.org/**, soit en employant leur éditeur/navigateur gratuit Amaya.



Emplacement

Un site Web est susceptible de changer d'emplacement. Ce peut être une décision de niveau serveur, dont vous n'êtes pas maître. Un truc tout simple pour tester ce problème consiste à changer votre site en construction de disque ou de répertoire. Vous verrez ainsi si cette modification pose problème.

Bref, testez, testez... il en restera toujours quelque chose.

N'oubliez jamais que le lecteur d'une présentation informatisée établira toujours une corrélation inconsciente entre la valeur du contenu de celle-ci et son aspect.



XHTML

Pour convertir rapidement un document au format XHTML, vous pouvez vous servir de HTML Tidy ou de l'éditeur/navigateur Amaya, qui peut enregistrer des documents HTML en XHTML.

Réactions du public

Vous avez conçu un splendide site Web, l'avez testé par tous les moyens possibles. Comment toutefois savoir avec quelle efficacité votre message parvient au public ciblé ? Pour ce faire, vous devez fournir aux utilisateurs un moyen d'entrer en contact avec vous.

Le premier moyen pour encourager vos visiteurs à vous faire part de leurs remarques consiste à placer sur chaque page un lien vers votre adresse électronique. Vous pouvez utilement compléter cela par un lien situé sur votre page d'accueil, qui précise où signaler un problème. Nous l'avons fait pour notre site, grâce au lien de messagerie de la barre de navigation (affichée sur chaque page) et au message qui figure dans la page d'accueil, même si pour le moment les liens de ce message ne possèdent pas de cible.

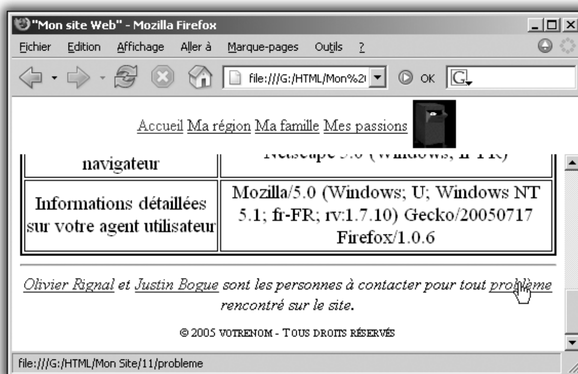


Figure 11.28 : *Il est capital de proposer un dispositif permettant aux utilisateurs de vous contacter*

Un autre moyen consiste à proposer un formulaire en ligne, qui peut être complété par les visiteurs et envoyé (probablement par messagerie) au responsable du site. Dans notre cas, ce serait une page *probleme.html*. Le formulaire doit poser des questions précises relatives au site, ainsi que sur le profil du visiteur. Ces dernières informations permettent de vérifier que le visiteur fait bien partie de la cible choisie, tandis que les questions portant sur le site permettent de focaliser les critiques éventuelles.

Des solutions plus avancées consistent à proposer une liste de diffusion ou une adresse de groupe de discussion. Une liste de diffusion envoie un message à toute personne présente sur la liste, les membres pouvant le plus souvent répondre directement à la liste. Cela peut augmenter la portée de votre site Web tandis que votre public pourra plus facilement vous faire part de ses remarques.

Un groupe de discussion est un endroit où des utilisateurs peuvent envoyer et lire des messages, regroupés par sujet ou par thème. Cela est très similaire à la liste de diffusion, mais attire traditionnellement moins de gens : sans doute parce que les utilisateurs doivent les visiter activement, alors qu'ils reçoivent automatiquement les messages d'une liste.

Ces deux méthodes demandent toutefois une mise en œuvre plus complexe, qui ne sera pas abordée ici.

Quelle que soit la façon dont vous encouragez les commentaires touchant votre site, répondez-y toujours rapidement. Si quelqu'un vous écrit pour vous demander une URL, envoyez-la le jour même ou répondez que vous ne savez pas où la trouver. N'hésitez jamais à avouer que vous ignorez quelque chose, mais essayez d'aller trouver la réponse.

11.5. Publication sur le Web

La dernière étape, et paradoxalement la plus simple, consiste à placer votre site sur le Web. Pour le moment, vous possédez un site en construction, stocké quelque part sur votre disque dur. Pour continuer, vous avez besoin d'un hébergeur et d'un outil capable de transférer les fichiers qui composent votre site sur le serveur de cet hébergeur.

Le problème se pose différemment pour une entreprise dotée de son propre réseau (et qui dispose probablement d'un ou de plusieurs

serveurs hébergeant son intranet et son site Web externe), pour une entreprise passant par l'intermédiaire d'un fournisseur d'accès Internet (FAI) spécialisé et pour un individu ou une PME pouvant se contenter des services d'un FAI grand public. C'est à ce dernier cas que nous nous intéressons particulièrement.

Peut-être l'ignorez-vous, mais si vous disposez d'un accès Internet, vous possédez probablement un hébergeur. La quasi-totalité des FAI mettent gracieusement à votre disposition un espace de stockage capable d'héberger vos pages personnelles. La place offerte se compte désormais en centaines de mégaoctets, un chiffre excédant en principe largement les besoins d'un site personnel ou SOHO (*Small Home, Home Office*). Orange propose par exemple 100 Mo d'espace pour le stockage de vos pages personnelles, Free montant jusqu'à 10 Go !

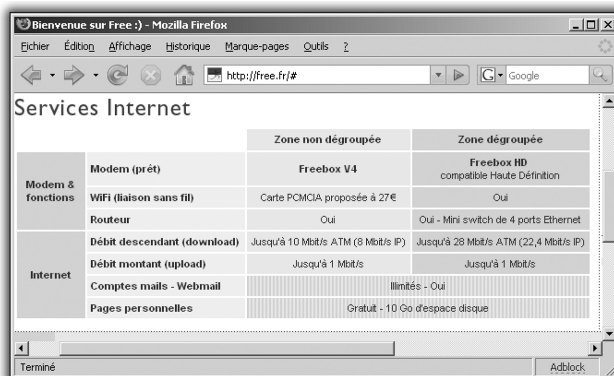


Figure 11.29 : *Gestion des pages personnelles de Free*

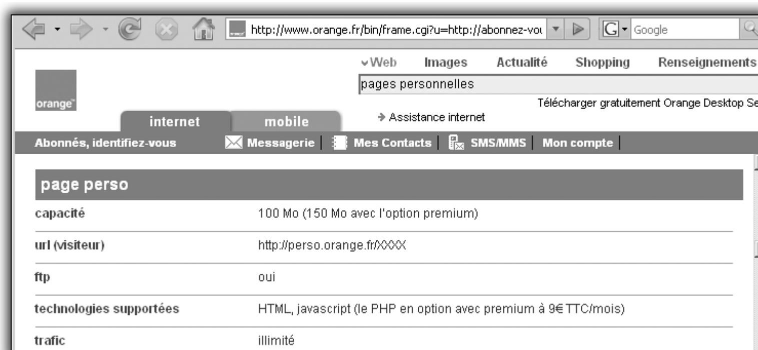


Figure 11.30 : *Offre standard Orange*

Certains FAI proposent, comme Orange, un hébergement amélioré pour un montant relativement modeste.

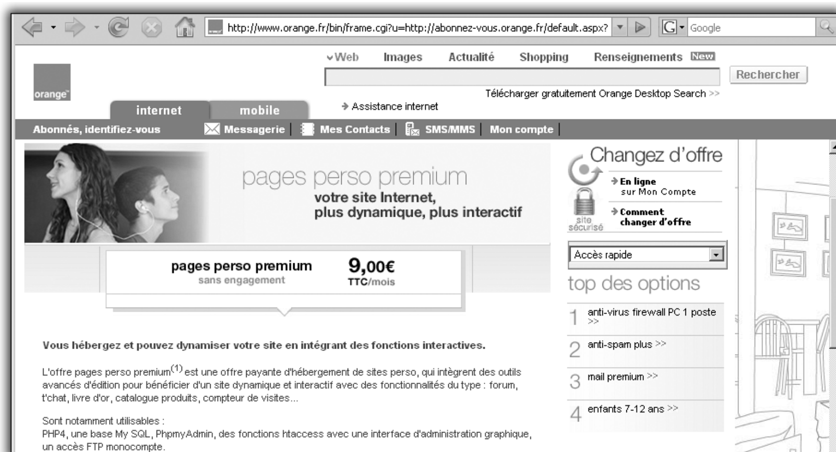


Figure 11.31 : *Hébergement amélioré d' Orange*

Vous devrez dans la plupart des cas activer vos pages personnelles (votre compte de pages personnelles) auprès de votre FAI. Suivez pour cela les instructions fournies par celui-ci dans sa documentation ou sur son site.

Vous devez ensuite disposer d'un moyen de transférer vos fichiers du répertoire de construction, situé sur votre disque dur, jusqu'au répertoire de destination, situé à une adresse fournie par votre FAI (souvent quelque chose comme `votrenom.perso-ftp`, `votreFAI.fr` ou `votrenom.perso.votreFAI.fr`). Cela s'effectue à l'aide d'un protocole nommé FTP (*File Transfer Protocole*), si bien que les logiciels capables d'accomplir ce travail portent le nom de « logiciels FTP ».

Certains éditeurs HTML proposent des fonctions FTP intégrées pour le transfert et l'administration de votre site Web. Si ce n'est pas le cas de votre éditeur favori, il vous reste, selon les FAI, deux solutions :

- **WebFTP** : cette méthode relativement récente proposée par certains FAI (dont Neuf-Telecom) est similaire au WebMail, ce système qui vous permet de consulter vos courriers sans logiciel de messagerie dédié, depuis n'importe quel navigateur. Généralement très intuitive, c'est une méthode élégante.

- **Logiciels FTP indépendants** : il en existe un grand nombre, certains gratuits, certains sous forme de partagiciels (*shareware*), d'autres enfin étant commerciaux. Votre FAI propose généralement dans sa boîte à outil un tel produit (Cuteftp pour Orange, LeechFtp sous Windows ou Fetch sous MacOS proposés sur le kit de connexion de Free, mais il en existe bien d'autres, comme vous le montrera une simple recherche Google). Citons notament Filezilla, de la grande famille des logiciels libres gratuits (**www.filezilla-project.org**).

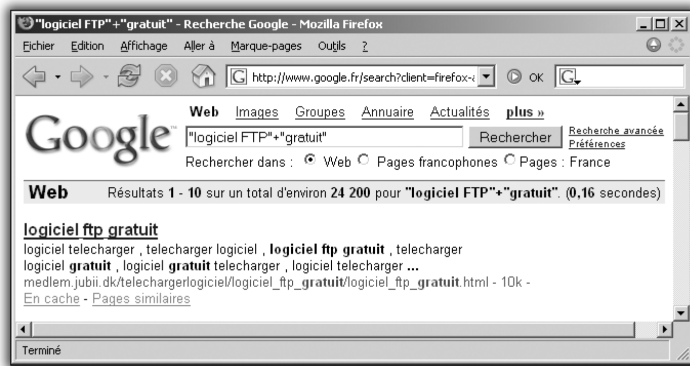


Figure 11.32 : *Une recherche Google avec comme mots-clés « logiciel FTP »+ « gratuit » procure plus de 6 600 réponses*

Si vous avez choisi de télécharger et d'installer un logiciel FTP, vous devez le configurer. Pour ce faire, vous devez toujours lui indiquer :

- L'adresse du serveur FTP. C'est généralement quelque chose comme perso-ftp.*votreFAI*.fr
- Votre identifiant et votre mot de passe de messagerie.
- L'emplacement de vos pages sur votre disque dur (le site local).

Ces informations vous sont fournies par votre FAI.

Selon le logiciel, d'autres informations peuvent vous être demandées :

- Le type d'hôte. Cette information vous est fournie par votre FAI. C'est souvent UNIX (standard).
- Un nom pour la connexion. Pratique si ce logiciel doit servir à vous connecter à plusieurs serveurs STP.

- L'adresse de votre site. C'est souvent quelque chose comme `http://perso.votreFAI.fr/votrenom/` ou `votrenom.perso.votreFAI.fr`.

Les copies d'écran qui suivent montrent les informations demandées par WS_FTP95 LE, mon logiciel FTP favori (bien qu'ancien).

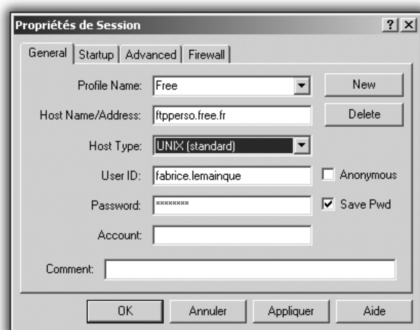


Figure 11.33 : *Informations de connexion demandées par WS_FTP95 LE*

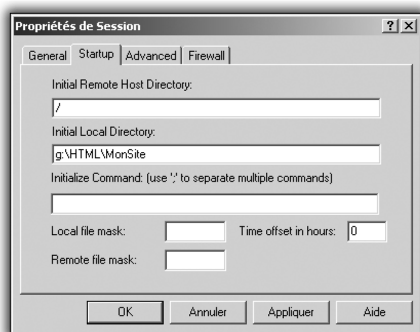


Figure 11.34 : *Informations de connexion demandées par WS_FTP95 LE*

Vous voyez en principe, dans la fenêtre du logiciel, deux volets. Le volet de gauche présente les fichiers situés dans le dossier spécifié lors de la configuration du logiciel. Si ce n'est pas le bon dossier, vous pouvez naviguer sur votre ordinateur jusqu'à afficher le dossier contenant les fichiers de votre site.

Établissez la connexion si elle ne l'est pas déjà (la méthode peut différer selon le logiciel). Le cadre de droite affiche les fichiers et dossiers actuellement situés sur le serveur distant : ceux effectivement consultés par des visiteurs.

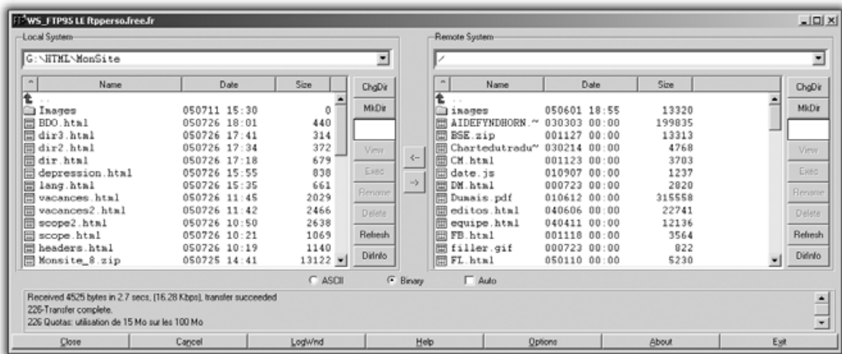


Figure 11.35 : *Fenêtre de WS_FTP95 LE, affichant les dossiers de MonSite et ceux situés sur le serveur distant.*

Pour transférer des fichiers, sélectionnez-les dans le volet de gauche et lancez le transfert, par exemple en cliquant sur un bouton.

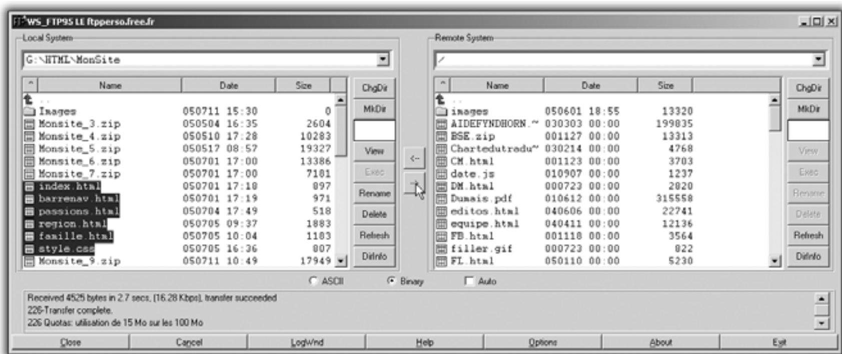


Figure 11.36 : *Sélection et transfert de fichiers*

Un message d'état du transfert des fichiers est généralement affiché.

Patientez quelques instants et allez visiter votre site, en lançant votre navigateur et en saisissant l'adresse de votre site.

Ce même logiciel permet de gérer votre site à distance. Vous pouvez à tout moment ajouter de nouveaux fichiers ou en supprimer. Pour supprimer un fichier distant, sélectionnez-le simplement dans le cadre de droite et cliquez sur **Supprimer** ou **Delete**.

11.6. Suivi de la fréquentation de votre site

Une fois votre site en ligne, vous souhaiterez probablement en suivre la fréquentation d'une façon ou d'une autre. De nombreux FAI proposent de tels outils, parfois gratuitement, parfois moyennant finances. Leur emploi est parfois simple, parfois complexe. Et tout changement de FAI impose d'étudier un nouveau système…

Une solution alternative consiste à recourir à Google Analytics, un outil gratuit d'une rare puissance. Voici comment en profiter :

1 Servez-vous de votre compte Google pour ouvrir un compte Google Analytics, à partir de la page **http://www.google.com/analytics/fr-FR/**.



Figure 11.37 : *Page d'accueil de Google Analytics.*

2 Dans la zone de texte *S'inscrire à Google Analytics*, cliquez sur **Inscription**.

3 Entrez l'URL du site Web que vous souhaitez suivre, puis nommez votre compte. C'est ce nom qui figurera dans vos rapports Google Analytics. Vous pourrez, si vous le souhaitez, ajouter d'autres sites après la configuration de votre compte.

4 Renseignez vos nom, prénom, numéro de téléphone et pays.

5 Acceptez le contrat utilisateur et cliquez sur *Créer un compte*. Cela affiche la page *Instructions de suivi*. Cette page possède une zone de texte qui contient un fragment de code. Dans celui-ci, `UA–xxxxxxx–x` est l'identifiant unique associé à votre compte:

```
<script src="http://www.google-analytics.com/urchin.js"
type="text/javascript">
</script>
<script type="text/javascript">
  _uacct = "UA-xxxxxxx-x";
  urchinTracker();
</script>
```

Ce fragment de code charge la bibliothèque *urchin.js* dans l'agent utilisateur client, définit l'identifiant unique (`_uacct`), puis appelle `urchinTracker()` pour renvoyer les données à Google Analytics.

Vous devez copier ce fragment de code dans chaque page Web que vous voulez suivre. Vous le faites en principe juste avant la balise de fermeture `</BODY>`, mais vous pouvez également le placer dans l'élément `HEAD`, surtout si votre page est riche en code JavaScript et que vous constatez quelques problèmes.



L'activation de Google Anaytics n'est pas immmédiate

L'arrivée des premières données dans vos rapports peut demander près de 24 heures après l'ouverture d'un compte et la saisie, puis la publication du code de suivi adéquat. Soyez patient…

Vous pouvez examiner les états associés à votre gadget en ouvrant une session dans Google Analytics et en choisissant **Afficher les rapports**.

La page qui s'affiche alors est le *Tableau de bord* (voir Figure 11.38).

Il présente six volets principaux.

- En haut, un graphique de l'évolution des visites quotidiennes ;

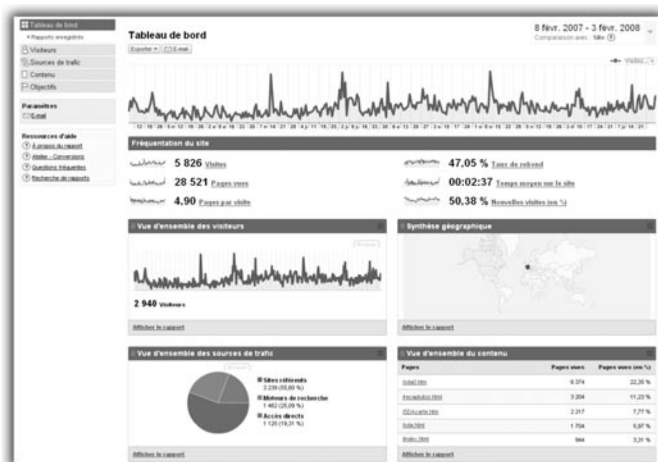


Figure 11.38 : *Tableau de bord Google Analytics.*

- En dessous, différentes données sur la **Fréquentation du site** (nombre de visites sur la période, pages vues, pages par visites, rapport nouveaux/anciens visiteurs etc.) ;
- Vient ensuite une **Vue d'ensemble des visiteurs**, très similaire au premier graphique si ce n'est qu'il concerne les visiteurs et non les visites, et une **Synthèse géographique** : une carte du monde où l'intensité de la couleur est proportionnelle au nombre de visiteurs de même d'origine géographique.
- Le dernier groupe présente une **Vue d'ensemble des sources de trafic (**un camembert qui montre par quel moyen les visiteurs sont parvenus à votre site) et une **Vue d'ensemble du contenu** (les pages les plus vues).

Vous pouvez également éventuellement voir un graphique de Vue d'ensemble des objectifs, si vous avez défini un objectif dans les paramètres d'analyse.

Il suffit de cliquer sur le lien *Afficher le rapport* d'un des volets ou sur un élément pour afficher ses détails. L'accès est également possible à partir du menu de gauche. Il est impossible de tout détailler : cela nécessiterait un ouvrage entier !

Intéressons-nous plutôt à l'élément *Visiteurs* : cliquez à gauche sur *Visiteurs* ou sur *Afficher le rapport* du volet **Vue d'ensemble des visiteurs**. Cela affiche une nouvelle page qui fourmille d'informations

intéressantes (*Segmentation des visiteurs*, *Profil des visiteurs dont langue(s)*, *Fournisseurs d'accès et Valeurs personnalisées*, *Type de navigateur* [navigateurs, systèmes d'exploitation, navigateur et systèmes d'exploitation, couleurs d'écran, résolutions d'écran, types de prise en charge de Java, Flash] et *Synthèse géographique*)

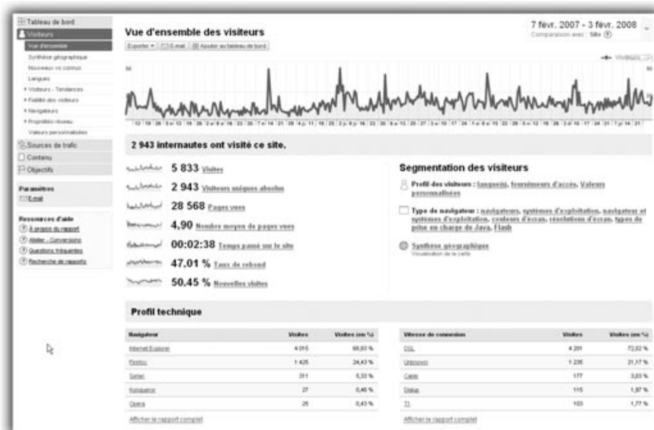


Figure 11.39 : *Page Vue d'ensemble des visiteurs*

Cliquons dans le volet *Navigateurs* de la section *Profil technique* sur **Afficher le rapport complet**. Cela peut permettre de vérifier certains postulats émis précédemment. Je précise toutefois que cela ne s'applique qu'à mon site, dont les 5833 visiteurs constatés sur environ 1 an ne constituent pas forcément un échantillon représentatif du Web. En revanche, c'est sans aucun doute "mon" public.



Figure 11.40 : *Navigateurs*

Mes visiteurs possèdent dans 68,8 % des cas Internet Explorer, et Firefox dans 24,4 % des cas. Vous pouvez cliquer sur le nom d'un navigateur pour voir des détails sur ses versions : ainsi, Internet Explorer 6 était encore employé dans 50,7 % des cas, soit par environ 35 % de mes visiteurs. Il n'existe quasiment plus de versions antérieures

(0,5 %). En revanche, les utilisateurs de Firefox semblent plus enclins à rester à jour, car 97 % d'entre eux possèdent au moins la version 2.0.0.1.

Cliquons ensuite dans le menu de gauche sur **Systèmes d'exploitation**.



Figure 11.41 : *Versions de la plate-forme.*

Leur système d'exploitation est Windows dans 92 % des cas. Il devient intéressant de regarder la combinaison navigateur/plate-forme en cliquant dans le menu de gauche sur Navigateurs et systèmes d'exploitation pour disposer d'une idée plus précise de leurs caractéristiques.

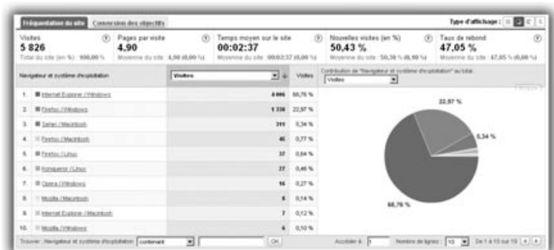


Figure 11.42 : *Combinaisons Navigateur et Plate-forme.*

Internet Explorer sur Windows 68,7 % de mes visiteurs, suivi de Firefox sur Windows (23 %). Safari sur Macintosh atteint les 5 %, suivi de Firefox toujours sur Macintosh (1 %). Les autres combinaisons restent anecdotiques, ne totalisant que 3 % des visiteurs.

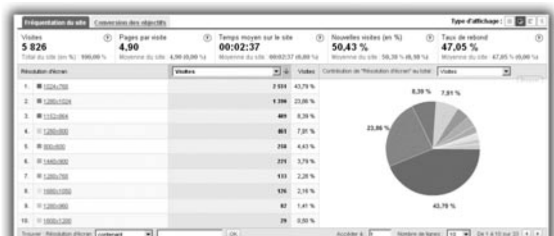


Figure 11.43 : *Résolutions d'écran.*

Seuls (mais encore) 4,4 % des visiteurs possèdent une résolution de 800 x 600 pixels. 43,7 % disposent d'une résolution de 1024 x 768, les autres possédant une résolution supérieure. Si je dois donc toujours prévoir une résolution minimale de 800 x 600, les choses risquent d'évoluer rapidement…

Ces informations sont précieuses : elles confirment que je ne puis me dispenser de tester mes pages à l'aide au moins d'Internet Explorer 6 et 7 et de Firefox 2.

Google Analytics apparaît donc comme un outil précieux, presque indispensable. Ne vous en privez pas !

11.7. Résumé

- Dès le début de la phase de conception de votre site, vous devez garder à l'esprit les caractéristiques de la cible visée et, pour ce faire, identifier votre public :
  - Les enfants préfèrent des images claires et colorées et sont souvent capables de comprendre le but d'une icône sans avoir à lire le texte.
  - Les personnes âgées peuvent souffrir de déficits visuels et éprouver des difficultés à employer le clavier ou la souris.
  - Un public d'amateurs de jeux vidéo apprécie particulièrement les images et les vidéos.
- Les cadres rendent la navigation difficile sans souris et ne sont pas reconnus par les navigateurs textuels.
- Les images posent rarement de problèmes si vous avez pensé à placer un attribut `alt` dans l'élément `IMG` pour permettre aux utilisateurs ayant désactivé les images d'en obtenir une description.
- Les tableaux sont assez mal reconnus par les navigateurs textuels. Servez-vous toujours du nouvel attribut `summary` pour procurer une alternative texte au tableau. Pour une cellule de données particulière, l'attribut `headers` répertorie les cellules qui fournissent des indications de rubrique pertinentes. Pour une cellule de rubrique donnée, l'attribut `scope` indique à l'agent utilisateur les cellules concernées par ces indications de rubrique. Enfin, l'attribut `abbr` spécifie un nom de rubrique abrégé pour les

cellules de rubrique, afin que les agents utilisateurs puissent restituer plus rapidement les indications de rubrique.

- Les jeux d'encadrement font également partie des structures délicates à interpréter par les agents utilisateurs non visuels. La seule solution consiste à employer un élément `NOFRAME` et à fournir un lien alternatif vers une version dépourvue de cadres.
- Les modules complémentaires (ou *plug-in*) sont de petits programmes qui permettent d'afficher dans le navigateur des types de contenu normalement non affichables. Les plus célèbres sont Adobe Acrobat Reader, Shockwave, JVM (*Java Virtual Machine*), QuickTime et RealAudio. Ces types de contenu sont insérés dans une page Web à l'aide de l'élément `OBJET`.
- L'ensemble de caractères Unicode désormais adopté par HTML dispose de plus de 25 000 caractères différents et est capable de représenter les langues du monde entier.
- Vous spécifiez la langue d'un élément HTML à l'aide de son attribut `lang`. Celui-ci reçoit comme valeur un code qui identifie une langue parlée, écrite, ou utilisée d'une manière ou d'une autre pour la communication d'informations entre personnes.
- L'attribut `dir` spécifie la direction de base d'un texte dépourvu de directionnalité inhérente telle que définie dans la norme Unicode, ainsi que la directionnalité des tableaux. Il ne semble toutefois pas être très reconnu par les navigateurs actuels.
- Il est possible de désactiver l'algorithme bidirectionnel Unicode et de spécifier une directionnalité à l'aide de l'élément `BDO` *(BiDirectional Override).* Cet élément possède un attribut obligatoire `dir`, identique au précédent, mais qui prévaut sur la directionnalité inhérente des caractères telle que définie dans Unicode.
- Si vous réalisez une page dans une langue autre qu'européenne, vous devez modifier le jeu de caractères du document à l'aide de l'attribut `charset` d'un élément `META`. De nombreux navigateurs permettent d'imposer le choix du type d'encodage de la page.
- Il faut toujours tester un site Web avec autant de navigateurs que possible, et au minimum sous Internet Explorer 5.0 et 6.0 et sous Firefox.
- Vous devez également effectuer un test avec ces navigateurs en désactivant l'affichage des images.

- Vous pouvez trouver sur le Net des sites de test, comme AnyBrowser.com. Vous pouvez y effectuer un test en spécifiant un niveau de compatibilité avec les spécifications HTML et même tester un affichage WebTV.
- Vous devez fournir aux utilisateurs un moyen d'entrer en contact avec vous et les encourager à vous faire part de leurs remarques.
- Pour placer effectivement votre site sur le Web, vous avez besoin d'un hébergeur et d'un outil capable de transférer les fichiers qui composent votre site sur le serveur de cet hébergeur. La quasi-totalité des FAI mettent gracieusement à votre disposition un espace de stockage capable d'héberger vos pages personnelles.
- Le transfert de vos fichiers du répertoire de construction jusqu'au répertoire de destination, situé chez l'hébergeur, s'effectue à l'aide d'un protocole nommé FTP (*File Transfer Protocole*). Les logiciels capables d'accomplir ce travail portent le nom de « logiciels FTP ». Il existe de nombreux logiciels FTP gratuits, chaque FAI disposant de ses propres préférences.
- Google Analytics est un outil précieux qui permet de suivre avec une grande précision la fréquentation de son site Web..

Chapitre 12

Annexes

Jeu des entités de caractères Latin-1 488
Éléments et attributs HTML 4.01 491
Compatibilité XHTML/HTML 513
DTD et XML 523
Sélecteurs CSS 530
Ressources Web 533

12.1. Jeu des entités de caractères Latin-1

Tableau 12.1 : *Jeu des entités de caractères Latin-1*			
Caractère	**Nom de l'entité**	**Codage décimal**	**Codage hexadécimal**
espace insécable	` `	` `	` `
¡	`¡`	`¡`	`¡`
¢	`¢`	`¢`	`¢`
£	`£`	`£`	`£`
¤ (symbole monétaire)	`¤`	`¤`	`¤`
¥	`¥`	`¥`	`¥`
¦	`¦`	`¦`	`¦`
§	`§`	`§`	`§`
-	`¨`	`¨`	`¨`
©	`©`	`©`	`©`
ª	`ª`	`ª`	`ª`
«	`«`	`«`	`«`
(signe négation)	`¬`	`¬`	`¬`
- (trait d'union virtuel)	``	``	``
®	`®`	`®`	`®`
¯ (macron)	`¯`	`¯`	`¯`
°	`°`	`°`	`°`
±	`±`	`±`	`±`
^{2}	`²`	`²`	`²`
^{3}	`³`	`³`	`³`
´	`´`	`´`	`´`

Tableau 12.1 : *Jeu des entités de caractères Latin-1*

Caractère	Nom de l'entité	Codage décimal	Codage hexadécimal
µ	µ	µ	µ
¶·(pied de mouche)	¶	¶	¶
·	·	·	·
¸	¸	¸	¸
^{1}	¹	¹	¹
º	º	º	º
»	»	»	»
¼	¼	¼	¼
½	½	½	½
¾	¾	¾	¾
¿	¿	¿	¿
À	À	À	À
Á	Á	Á	Á
Â	Â	Â	Â
Ã	Ã	Ã	Ã
Ä	Ä	Ä	Ä
Å	Å	Å	Å
Æ	Æ	Æ	Æ
Ç	Ç	Ç	Ç
È	È	È	È
É	É	É	É
Ê	Ê	Ê	Ê
Ë	Ë	Ë	Ë
Ì	Ì	Ì	Ì
Í	Í	Í	Í
Î	Î	Î	Î

Caractère	Nom de l'entité	Codage décimal	Codage hexadécimal
Ï	Ï	Ï	Ï
Ð	Ð	Ð	Ð
Ñ	Ñ	Ñ	Ñ
Ò	Ò	Ò	Ò
Ó	Ó	Ó	Ó
Ô	Ô	Ô	Ô
Õ	Õ	Õ	Õ
Ö	Ö	Ö	Ö
×	×	×	×
Ø	Ø	Ø	Ø
Ù	Ù	Ù	Ù
Ú	Ú	Ú	Ú
Û	Û	Û	Û
Ü	Ü	Ü	Ü
Ý	Ý	Ý	Ý
Þ	&THORN ;	Þ	Þ
ß	&szlig ;	ß	ß
à	à	à	à
á	á	á	á
â	â	â	â
ã	ã	ã	ã
ä	ä	ä	ä
å	å	å	å
æ	æ	æ	æ
ç	ç	ç	ç
è	è	è	è
é	é	é	é

Tableau 12.1 : *Jeu des entités de caractères Latin-1*

Tableau 12.1 : Jeu des entités de caractères Latin-1			
Caractère	Nom de l'entité	Codage décimal	Codage hexadécimal
ê	&ecirc ;	ê	ê
ë	ë	ë	ë
ì	ì	ì	ì
í	í	í	í
î	î	î	î
ï	ï	ï	ï
∂	ð	ð	ð
ñ	ñ	ñ	ñ
ò	ò	ò	ò
ó	ó	ó	ó
ô	&ocirc ;	ô	ô
õ	õ	õ	õ
ö	ö	ö	ö
÷	÷	÷	÷
ø	ø	ø	ø
ù	ù	ù	ù
ú	ú	ú	ú
û	û	û	ྴ
ü	ü	ü	ü
ý	ý	ý	ý
þ	þ	þ	þ
ÿ	ÿ	ÿ	ÿ

12.2. Éléments et attributs HTML 4.01

Vous trouverez ici la totalité des éléments et attributs proposés par la spécification HTML 4.01, classés par ordre alphabétique.

Éléments HTML 4.01

A : ancre
`accesskey` : touche d'accès rapide. `%Character;`
`charset` : jeu de caractères de la ressource liée. `%Charset;`
`coords` : employé avec les cartes côté client. `%Coords;`
`href` : URI de la ressource liée. `%URI;`
`hreflang` : code de la langue. `%LanguageCode;`
`name` : fin de lien nommé. `CDATA`
`onblur` : l'élément a perdu le focus. `%Script;`
`onfocus` : l'élément a obtenu le focus. `%Script;`
`rel` : types de liens avant. `%LinkTypes;`
`rev` : types de liens arrière. `%LinkTypes;`
`shape` : employé avec les cartes côté client. `%Shape;`
`tabindex` : position dans l'ordre de tabulation. `NUMBER`
`target` : afficher dans ce cadre (DTD Frameset). `%FrameTarget;`
`type` : type de contenu. `%ContentType;`

ABBR : abréviation (`www`, `HTTP`, etc.)

ACRONYM : acronyme

ADDRESS : informations sur l'auteur

APPLET : applet Java (déconseillé, DTD Transitoire)
`align` : alignement vertical ou horizontal (déconseillé). `%IAlign;`
`alt` : description brève (déconseillé). `%Text;`
`archive` : liste d'archives, séparées par des virgules (déconseillé). `CDATA`
`code` : fichier de classes de l'applet (déconseillé). `CDATA`
`codebase` : URI facultatif de base pour l'applet (déconseillé). `%URI;`
`height` : hauteur initiale (déconseillé). `%Length;`
`hspace` : intervalle horizontal (déconseillé). `%Pixels;`
`name` : permet aux applets de se retrouver mutuellement (déconseillé). `CDATA`
`object` : fichier d'applet sérialisé (déconseillé). `CDATA`
`vspace` : intervalle vertical (déconseillé). `%Pixels;`
`width` : largeur initiale (déconseillé). `%Length;`

AREA : image cliquable côté client
`accesskey` : touche d'accès rapide. `%Character;`
`alt` : description brève. `%Text;`
`coords` : liste de longueurs séparées par des virgules. `%Coords;`

`href` : URI de la ressource liée. `%URI;`
`nohref` : cette région est sans action. `(nohref)`
`onblur` : l'élément a perdu le focus. `%Script;`
`onfocus` : l'élément a obtenu le focus. `%Script;`
`shape` : contrôle l'interprétation des coordonnées. `%Shape;`
`tabindex` : place dans l'ordre de tabulation. `NUMBER`
`target` : afficher dans ce cadre (DTD Frameset). `%FrameTarget;`

B : texte en style gras

BASE : URI de base du document
`href` : URI se comportant comme URI de base. `%URI;`
`target` : afficher dans ce cadre (DTD Frameset). `%FrameTarget;`

BASEFONT : taille de la police de base (déconseillé, DTD transitoire)
`color` : couleur du texte (déconseillé). `%Color;`
`face` : liste de noms de police, séparés par des virgules (déconseillé). `CDATA`
`size` : obligatoire. Taille de police de base des éléments `FONT` (déconseillé). `CDATA`

BDO : inactivation de l'algorithme I18N BiDi
`dir` : direction (obligatoire). `(ltr | rtl)`

BIG : texte agrandi

BLOCKQUOTE : citation longue
`cite` : URI du document source ou message. `%URI;`

BODY : corps du document
`alink` : couleur des liens sélectionnées (déconseillé). `%Color;`
`background` : image d'arrière-plan du document (déconseillé). `%URI;`
`bgcolor` : couleur d'arrière-plan du document (déconseillé). `%Color;`
`link` : couleur des liens (déconseillé). `%Color;`
`onload` : le document a été chargé. `%Script;`
`onunload` : le document a été supprimé. `%Script;`
`text` : couleur du texte du document (déconseillé). `%Color;`
`vlink` : couleur des liens visités (déconseillé). `%Color;`

BR : saut de ligne
`clear` : contrôle du flux de texte (déconseillé). `(left | all | right | none)`

BUTTON : bouton
`accesskey` : touche d'accès rapide. `%Character;`
`disabled` : non disponible dans ce contexte. `(disabled)`
`name` : nom. `CDATA`
`onblur` : l'élément a perdu le focus. `%Script;`
`onfocus` : l'élément a obtenu le focus. `%Script;`
`tabindex` : position dans l'ordre de tabulation. `NUMBER`
`type` : employé comme bouton de formulaire. `(button | submit | reset)`
`value` : valeur envoyée au serveur lors de la soumission du formulaire. `CDATA`

CAPTION : légende de tableau
`align` : alignement relatif du tableau (déconseillé). `%CAlign;`

CENTER : raccourci pour `DIV align="center"` (déconseillé, DTD transitoire)

CITE : citation

CODE : fragment de code informatique

COL : colonne de tableau
`align` : alignement (déconseillé). `(left | center | right | justify | char)`
`char` : caractère d'alignement, par exemple `char=':'.%Character;`
`charoff` : décalage du caractère d'alignement. `%Length;`
`span` : les attributs de `COL` affectent `n` colonnes. `n (nombre)`
`valign` : alignement vertical dans les cellules. `(top | middle | bottom | baseline)`
`width` : spécification de la largeur des colonnes. `%MultiLength;`

COLGROUP : groupe de colonnes de tableau
`align` : alignement (déconseillé). `(left | center | right | justify | char)`
`char` : caractère d'alignement, par exemple `char=':'. %Character;`
`charoff` : décalage du caractère d'alignement. `%Length;`
`span` : nombre de colonnes par défaut dans un groupe. `NUMBER`
`width` : largeur par défaut des `COL` concernées (imbriquées). `%MultiLength;`
`valign` : alignement vertical dans les cellules. `(top | middle | bottom | baseline)`

DD : description de définition

DEL : texte supprimé
cite : information sur le motif de la modification.
datetime : date et heure de modification. %Datetime;

DFN : définition d'instance

DIR : liste de répertoire (déconseillé, DTD transitoire)
compact : réduction de l'espace entre éléments (déconseillé). (compact)

DIV : conteneur de langue/style générique
align : alignement du texte (déconseillé). (left | center | right | justify)

DL : liste de définitions
compact : réduction de l'espace entre éléments (déconseillé). (compact)

DT : terme de définition

EM : emphase

FIELDSET : groupe de contrôles de formulaire

FONT : modification(s) locale(s) d'une police (déconseillé, DTD transitoire)
color : couleur du texte (déconseillé). %Color;
face : liste de noms de police, séparés par une virgule (déconseillé). CDATA
size : par exemple size="+1", size="4" (déconseillé). [+|–]nn,

FORM : formulaire interactif
accept–charset : liste des jeux de caractères pris en charge. %Charsets;
accept : liste des types MIME pour le téléchargement de fichiers. %ContentTypes;
action : mode de gestion des formulaires côté serveur (obligatoire). %URI;
enctype : type d'encodage. %ContentType;
method : méthode HTTP employée pour soumettre le formulaire. (GET | POST)

`name` : nom du formulaire (employé dans des scripts). `CDATA`
`onreset` : le formulaire a été réinitialisé. `%Script;`
`onsubmit` : le formulaire a été soumis. `%Script;`
`target` : afficher dans ce cadre . `%FrameTarget;`

FRAME : sous-cadre (fenêtre secondaire) (DTD Frameset)
`longdesc` : lien vers une description longue, complétant le titre. `%URI;`
`marginheight` : marge haute en pixels. `%Pixels;`
`marginwidth` : marge basse en pixels. `%Pixels;`
`noresize` : l'utilisateur peut-il modifier la taille des cadres ? `(noresize)`
`frameborder` : présence ou absence de bordure pour le cadre. `1 | 0)`
`name` : nom du cadre pour pouvoir le cibler. `CDATA`
`scrolling` : présence ou absence des barres de défilement du cadre. `(yes | no | auto)`
`src` : source du contenu du cadre. `%URI;`

FRAMESET : subdivision de fenêtre (DTD Frameset)
`cols` : liste de longueurs, par défaut 100 % (1 colonne) (DTD Frameset). `%MultiLengths;`
`onload` : tous les cadres ont été chargés (DTD Frameset). `%Script;`
`onunload` : tous les cadres ont été supprimés (DTD Frameset). `%Script;`
`rows` : liste de longueurs, par défaut 100 % (1 ligne) (DTD Frameset). `%MultiLengths;`

H*n* : titre de niveau *n* (*n* compris entre 1 et 6)
`align` : alignement du texte (déconseillé). (left | center | right | justify)

HEAD : en-tête de document
`profile` : dictionnaire nommé de méta-informations. `%URI;`

HR : ligne horizontale
`align` : alignement du texte (déconseillé). `(left | center | right)`
`noshade` : ombrage (déconseillé). `(noshade)`
`size` : taille en pixels (déconseillé). `%Pixels;`
`width` : largeur (déconseillé). `%Length;`

HTML : élément racine du document
`version` : constante (déconseillé). `%HTML Version;`

I : texte de style italique

IFRAME : sous-fenêtre en ligne (DTD Transitoire)
`align` : alignement vertical ou horizontal (déconseillé). `%IAlign;`
`height` : hauteur du cadre. `%Length;`
`longdesc` : lien vers une description longue, complétant le titre. `%URI;`
`marginheight` : hauteur des marges en pixels. `%Pixels;`
`marginwidth` : largeur des marges en pixels. `%Pixels;`
`width` : largeur du cadre. `%Length;`
`frameborder` : présence ou absence de bordure de cadre. `(1 | 0)`
`name` : nom du cadre pour pouvoir le cibler. `CDATA`
`scrolling` : présence ou absence des barres de défilement du cadre. `(yes | no | auto)`
`src` : source du contenu du cadre. `%URI;`

IMG : image incorporée
`align` : alignement vertical ou horizontal (déconseillé). `%IAlign;`
`alt` : description brève (recommandé). `%Text;`
`border` : largeur de la bordure du lien (déconseillé). `%Pixels;`
`ismap` : emploi de carte image côté serveur. `(ismap)`
`height` : hauteur d'outrepassement. `%Length;`
`hspace` : décalage horizontal (déconseillé). `%Pixels;`
`longdesc` : lien vers une description longue (complétant `alt`).
`name` : nom de l'image (pour un script). `CDATA`
`src` : URI de l'image à incorporer (obligatoire). `%URI;`
`usemap` : emploi de carte image côté client. `%URI;`
`vspace` : décalage vertical (déconseillé). `%Pixels;`
`width` : largeur d'outrepassement. `%Length;`

INPUT : contrôle de formulaire (saisie)
`accept` : liste des types MIME pour le téléchargement de fichiers. `%ContentTypes;`
`accesskey` : touche d'accès rapide. `%Character;`
`align` : alignement vertical ou horizontal (déconseillé). `%IAlign;`
`alt` : description brève.
`checked` : pour les boutons d'option et les cases à cocher. `(checked)`
`disabled` : indisponible dans ce contexte. `(disabled)`
`ismap` : emploi de carte image côté serveur. `(ismap)`
`maxlength` : nombre maximum de caractères d'un champ texte. `NUMBER`
`name` : soumission comme partie d'un formulaire. `CDATA`
`onblur` : l'élément a perdu le focus. `%Script;`
`onchange` : la valeur de l'élément a été modifiée. `%Script;`
`onfocus` : l'élément a obtenu le focus. `%Script;`

`onselect` : du texte a été sélectionné. `%Script;`
`readonly` : pour du texte et un mot de passe. `(readonly)`
`size` : spécifique à chaque type de champ. `CDATA`
`src` : pour les champs à image. `%URI;`
`tabindex` : position dans l'ordre de tabulation. `NUMBER`
`type` : le type de widget nécessaire. `%InputType;`
`usemap` : emploi d'une carte image côté client. `%URI;`
`value` : spécifique aux boutons d'option et cases à cocher. `CDATA`

INS : texte inséré
`cite` : information sur le motif de la modification. `%URI;`
`datetime` : date et heure de la modification. `%Datetime;`

ISINDEX : invite à ligne unique (déconseillé, DTD transitoire)
`prompt` : message d'invite (déconseillé). `%Text;`

KBD : texte devant être saisi par l'utilisateur

LABEL : libellé d'un champ de formulaire
`accesskey` : touche d'accès rapide. `%Character;`
`for` : correspond à la valeur d'`ID` d'un champ.
`onblur` : l'élément a perdu le focus. `%Script;`
`onfocus` : l'élément a obtenu le focus. `%Script;`

LEGEND : légende d'un ensemble de champs
`accesskey` : touche d'accès rapide. `%Character;`
`align` : alignement relatif d'un ensemble de champs (déconseillé). `%LAlign;`

LI : élément de liste
`type` : style d'un élément de liste (déconseillé). `%LIStyle;`
`value` : réinitialisation du numéro de séquence (déconseillé). `NUMBER`

LINK : lien indépendant du média
`charset` : jeu de caractères de la ressource liée. `%Charset;`
`href` : URI de la ressource liée. `%URI;`
`media` : pour l'affichage sur un média précis. `%MediaDesc;`
`target` : afficher dans ce cadre (DTD Frameset). `%FrameTarget;`
`hreflang` : code de langue. `%LanguageCode;`
`rel` : type de lien avant. `%LinkTypes;`
`rev` : type de lien arrière. `%LinkTypes;`
`type` : type de contenu indicatif. `%ContentType;`

MAP : image cliquable côté client
`name` : servant de référence à `usemap` (obligatoire). `CDATA`

MENU : liste de menu (déconseillé, DTD transitoire)
`compact` : espace entre éléments réduit (déconseillé). `(compact)`

META : méta-information générique
`content` : information associée (obligatoire). `CDATA`
`http-equiv` : nom de l'en-tête de réponse HTTP. `NAME`
`name` : nom de la méta-information. `NAME`
`scheme` : sélection d'une forme de contenu. `CDATA`

NOFRAMES : conteneur de contenu alternatif pour un affichage non fondé sur des cadres (DTD Frameset)

NOSCRIPT : conteneur de contenu alternatif pour un affichage non fondé sur des scripts

OBJECT : objet incorporé générique
`align` : alignement vertical ou horizontal (déconseillé). `%IAlign;`
`archive` : liste d'URI séparées par des espaces. `CDATA`
`border` : largeur de la bordure du lien (déconseillé). `%Pixels;`
`classid` : identifie une mise en œuvre. `%URI;`
`codebase` : URI de base pour `classid`, `data`, `archive`. `%URI;`
`codetype` : type de contenu pour `code`. `%ContentType;`
`data` : référence vers les données de l'objet. `%URI;`
`declare` : déclaration d'un drapeau sans l'initialiser. `(declare)`
`height` : hauteur d'outrepassement. `%Length;`
`hspace` : intervalle horizontal (déconseillé). `%Pixels;`
`name` : soumettre comme partie d'un formulaire. `CDATA`
`standby` : message à afficher lors du chargement. `%Text;`
`tabindex` : position dans l'ordre de tabulation. `NUMBER`
`type` : type de contenu pour `data`. `%ContentType;`
`usemap` : emploi d'une carte image côté client. `%URI;`
`vspace` : intervalle vertical (déconseillé). `%Pixels;`
`width` : largeur d'outrepassement. `%Length;`

OL : liste ordonnée
`compact` : réduction de l'intervalle entre éléments (déconseillé). `(compact)`
`start` : début du numéro de séquence (déconseillé). `NUMBER`
`type` : style de numérotation (déconseillé). `%OLStyle;`

OPTGROUP : groupe d'options
`disabled` : indisponible dans ce contexte. `(disabled)`
`label` : employé dans les menus hiérarchiques (obligatoire). `%Text;`

OPTION : option pouvant être sélectionnée
`disabled` : indisponible dans ce contexte. `(disabled)`
`label` : employé dans les menus hiérarchiques. `%Text;`
`selected` : option présélectionnée. `(selected)`
`value` : par défaut, le contenu de l'élément. `CDATA`

P : paragraphe
`align` : alignement du texte (déconseillé). `(left | center | right | justify)`

PARAM : valeur de propriété nommée
`name` : nom de la propriété (obligatoire). `CDATA`
`type` : type de contenu pour la valeur lorsque `valuetype=ref`. `%ContentType;`
`value` : valeur de la propriété. `CDATA`
`valuetype` : mode d'interprétation de la valeur. `(DATA | REF | OBJECT)`

PRE : texte préformaté
`width` : (déconseillé). `NUMBER`

Q : courte citation en ligne
`cite` : URI du document source ou message. `%URI;`

S : texte de style barré (déconseillé, DTD transitoire)

SAMP : exemple de sortie de programme, scripts, etc.

SCRIPT : instructions de script
`charset` : jeu de caractères de la ressource liée. `%Charset;`
`defer` : l'agent utilisateur peut interrompre l'exécution du script. `(defer)`
`language` : nom de langage de script prédéfini (déconseillé). `CDATA`
`src` : URI d'un script externe. `%URI;`
`type` : type de contenu du langage de script (obligatoire). `%ContentType;`

SELECT : sélecteur d'option
`disabled` : indisponible dans ce contexte. `(disabled)`

`multiple` : la valeur par défaut est une sélection simple. `(multiple)`
`name` : nom du champ. `CDATA`
`onblur` : l'élément a perdu le focus. `%Script;`
`onfocus` : l'élément a obtenu le focus. `%Script;`
`size` : lignes visibles. `NUMBER`
`tabindex` : position dans l'ordre de tabulation. `NUMBER`
`onchange` : la valeur de l'élément a été modifiée. `%Script;`

SMALL : texte en style diminué

SPAN : conteneur de langue/style générique

STRIKE : texte barré (déconseillé, DTD transitoire)

STRONG : emphase forte

STYLE : informations de style
`media` : conçu pour l'emploi avec ces médias. `%MediaDesc;`
`type` : type de contenu du langage de style (obligatoire). `%ContentType;`

SUB : indice

SUP : exposant

TABLE : tableau
`align` : position relative du tableau dans la fenêtre (déconseillé). `%TAlign;`
`bgcolor` : couleur d'arrière-plan des cellules. `%Color;`
`border` : contrôle la largeur de la bordure autour du tableau. `%Pixels;`
`cellpadding` : espacement entre cellules. `%Length;`
`cellspacing` : espacement entre cellules. `%Length;`
`frame` : les parties du cadre à afficher. `%TFrame;`
`rules` : lignes entre lignes et colonnes. `%TRules;`
`summary` : but/structure pour une sortie parlée. `%Text;`
`width` : largeur du tableau. `%Length;`

TBODY : corps de tableau
`align` : alignement (déconseillé). `(left | center | right | justify | char)`
`char` : caractère d'alignement, par exemple `char=':'`. `%Character;`

`charoff` : décalage pour le caractère d'alignement. `%Length;`
`valign` : alignement vertical dans les cellules. (top | middle | bottom | baseline)

TD : cellule de données de tableau
`abbr` : abréviation pour les cellules d'en-tête. `%Text;`
`align` : alignement (déconseillé). `(left | center | right | justify | char)`
`axis` : liste séparée par des virgules d'en-têtes apparentés.
`bgcolor` : couleur d'arrière-plan d'une cellule (déconseillé). `%Color;`
`char` : caractère d'alignement, par exemple `char=':'`. `%Character;`
`charoff` : décalage pour le caractère d'alignement. `%Length;`
`colspan` : nombre de colonnes fusionnées. `NUMBER`
`headers` : liste d'`ID` des cellules d'en-tête. `IDREFS`
`height` : hauteur de la cellule (déconseillé). `%Length;`
`nowrap` : suppression du retour à la ligne automatique (déconseillé). `(nowrap)`
`rowspan` : nombre de lignes fusionnées dans une cellule. `NUMBER`
`scope` : portée des cellules d'en-tête. `%Scope;`
`valign` : alignement vertical dans les cellules. `(top | middle | bottom | baseline)`
`width` : largeur de la cellule (déconseillé). `%Length;`

TEXTAREA : champ texte à lignes multiples
`accesskey` : touche d'accès rapide. `%Character;`
`cols` : nombre de colonnes (obligatoire). `NUMBER`
`disabled` : indisponible dans ce contexte. `(disabled)`
`name` : nom. `#IMPLIED`
`onblur` : l'élément a perdu le focus. `%Script;`
`onchange` : la valeur de l'élément a été modifiée. `%Script;`
`onfocus` : l'élément a obtenu le focus. `%Script;`
`onselect` : du texte a été sélectionné. `%Script;`
`readonly` : lecture seule. `(readonly)`
`rows` : nombre de lignes (obligatoire). `NUMBER`
`tabindex` : position dans l'ordre de tabulation. `NUMBER`

TFOOT : pied de tableau
`align` : alignement (déconseillé). `(left | center | right | justify | char)`
`char` : caractère d'alignement, par exemple `char=':'`. `%Character;`
`charoff` : décalage pour le caractère d'alignement. `%Length;`
`valign` : alignement vertical dans les cellules. `(top | middle | bottom | baseline)`

TH : cellule d’en-tête de tableau
`abbr` : abréviation pour les cellules d’en-tête. `%Text;`
`align` : alignement (déconseillé). `(left | center | right | justify | char)`
`axis` : liste séparée par des virgules d’en-têtes apparentés. `CDATA`
`bgcolor` : couleur d’arrière-plan d’une cellule (déconseillé). `%Color;`
`char` : caractère d’alignement, par exemple `char=':'`. `%Character;`
`charoff` : décalage pour le caractère d’alignement. `%Length;`
`colspan` : nombre de colonnes fusionnées par cellule. `NUMBER`
`headers` : liste des `ID` des cellules d’en-tête. `IDREFS`
`height` : hauteur des cellules (déconseillé). `%Length;`
`nowrap` : suppression du passage automatique à la ligne (déconseillé). `(nowrap)`
`rowspan` : nombre de lignes fusionnées par cellule. `NUMBER`
`scope` : portée des cellules d’en-têtes. `%Scope;`
`valign` : alignement vertical dans les cellules. `(top | middle | bottom | baseline)`
`width` : largeur des cellules (déconseillé). `%Length;`

THEAD : en-tête de tableau
`align` : alignement (déconseillé). `(left | center | right | justify | char)`
`char` : caractère d’alignement, par exemple `char=':'`. `%Character;`
`charoff` : décalage pour le caractère d’alignement. `%Length;`
`valign` : alignement vertical dans les cellules. `(top | middle | bottom | baseline)`

TITLE : titre du document

TR : ligne de tableau
`align` : alignement (déconseillé). `(left | center | right | justify | char)`
`abbr` : abréviation pour les cellules d’en-tête. `%Text;`
`bgcolor` : couleur d’arrière-plan pour une ligne (déconseillé). `%Color;`
`char` : caractère d’alignement, par exemple `char=':'`. `%Character;`
`charoff` : décalage du caractère d’alignement. `%Length;`
`valign` : alignement vertical dans les cellules. `(top | middle | bottom | baseline)`

TT : texte de style téléscripteur, à espacement fixe

U : texte souligné (déconseillé, DTD transitoire)

UL : liste non ordonnée (liste à puces)
compact : réduction de l'espace entre éléments (déconseillé). (compact)
type : style de puces (déconseillé). %ULStyle;

VAR : instance de variable ou argument de programme

Attributs HTML 4.01

A
abbr : Abréviation pour cellule de rubrique. %Text;. Éléments TD et TH.
accept-charset : liste des jeux de caractères reconnus. %Charsets;. Élément FORM.
accept : liste de types MIME pour chargement d'un fichier sur le serveur. %ContentTypes;. Éléments FORM, INPUT.
accesskey : touche d'accès rapide %Character;. Éléments A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA.
action : Obligatoire. Gestionnaire de formulaires côté serveur.%URI;. Élément FORM.
align : Alignement relatif au tableau. %CAlign;. Élément CAPTION.
align : alignement vertical ou horizontal (déconseillé, DTD transitoire). %IAlign;. Éléments APPLET, IFRAME, IMG, INPUT, OBJECT.
align : relativement au jeu de champs (déconseillé, DTD transitoire). %LAlign;. Élément LEGEND.
align : position du tableau relativement à la fenêtre (déconseillé, DTD transitoire). %TAlign;. Élément TABLE.
align : alignement (déconseillé, DTD transitoire). (left | center | right). Élément HR.
align : alignement du texte (déconseillé, DTD transitoire). (left | center | right | justify). Éléments DIV, H1, H2, H3, H4, H5, H6, P.
align : alignement (déconseillé). (left | center | right | justify | char). Éléments COL, COLGROUP, TBODY, TD, TFOOT, TH, THEAD, TR.
alink : couleur des liens sélectionnés (déconseillé, DTD transitoire). %Color;. Élément BODY.
alt : brève description (déconseillé, DTD transitoire). %Text;. Élément APPLET.
alt : brève description. %Text;. Éléments AREA, IMG.

alt : brève description. CDATA. Élément INPUT.
archive : liste d'archives séparées par des virgules (déconseillé, DTD transitoire). CDATA. Élément APPLET.
archive : liste d'URI séparés par des espaces. CDATA. Élément OBJECT.
axis : liste des rubriques concernées séparées par des virgules. CDATA. Éléments TD, TH.

B

background : mosaïque de textures pour l'arrière-plan du document (déconseillé, DTD transitoire). %URI;. Élément BODY.
bgcolor : couleur d'arrière-plan des cellules (déconseillé, DTD transitoire). %Color;. Élément TABLE.
bgcolor : couleur d'arrière-plan des rangées (déconseillé, DTD transitoire). %Color;. Élément TR.
bgcolor : couleur d'arrière-plan des cellules (déconseillé, DTD transitoire). %Color;. Éléments TD,TH.
bgcolor : couleur d'arrière-plan du document (déconseillé, DTD transitoire). %Color;. Élément BODY.
border : épaisseur du contour du tableau. %Pixels;. Élément TABLE.
border : épaisseur de la bordure du lien (déconseillé, DTD transitoire). %Pixels;. Éléments IMG, OBJECT.

C

cellpadding : espacement à l'intérieur des cellules. %Length;. Élément TABLE.
cellspacing : espacement entre les cellules. %Length;. Élément TABLE.
char : caractère d'alignement, par exemple char=':'. %Character;. Éléments COL, COLGROUP, TBODY, TD, TFOOT, TH, THEAD, TR.
charoff : décalage du caractère d'alignement. %Length;. Éléments COL, COLGROUP, TBODY, TD, TFOOT, TH, THEAD, TR.
charset : encodage de caractères de la ressource reliée. %Charset;. Éléments A, LINK, SCRIPT.
checked : pour les boutons radio et les cases à cocher. (checked). Élément INPUT.
cite : URI du document ou message source. %URI;. Éléments BLOCKQUOTE, Q.
cite : informations sur la raison du changement. %URI;. Éléments DEL, INS.
class : liste de classes séparées par des espaces. CDATA. Tous les éléments sauf BASE, BASEFONT, HEAD, HTML, META, PARAM, SCRIPT, STYLE et TITLE.

classid : identifie une implémentation %URI;. Élément OBJECT.
clear : contrôle l'écoulement du texte (déconseillé, DTD transitoire). (left | all | right | none). Élément BR.
code : fichier de classe de l'applet (déconseillé, DTD transitoire). CDATA. Élément APPLET.
codebase : URI de base pour les attributs classid, data, archive. %URI;. Élément OBJECT.
codebase : URI de base optionnel pour l'applet (déconseillé, DTD transitoire). %URI;. Élément APPLET.
codetype : type de contenu de l'attribut code. %ContentType;. Élément OBJECT.
color : couleur du texte (déconseillé, DTD transitoire). %Color;. Éléments BASEFONT, FONT.
cols : liste de longueurs, par défaut : 100% (1 colonne) (DTD Frameset). %MultiLengths;. Élément FRAMESET.
cols : nombre de colonnes (obligatoire) (déconseillé, DTD transitoire). NUMBER. Élément TEXTAREA.
colspan : nombre de colonnes couvertes par la cellule. NUMBER. Éléments TD, TH.
compact : espacement inter-items réduit (déconseillé, DTD transitoire). (compact). Éléments DIR, DL, MENU, OL, UL.
content : informations associées (obligatoire). CDATA. Élément META.
coords : liste de longueurs séparées par des virgules. %Coords;. Élément AREA.
coords : à utiliser avec les images cliquables côté client. %Coords;. Élément A.

D

data : référence aux données de l'objet. %URI;. Élément OBJECT.
datetime : date et heure du changement. %Datetime;. Éléments DEL, INS.
declare : déclare mais n'instancie pas le drapeau. (declare). Élément OBJECT.
defer : l'agent utilisateur peut différer l'exécution du script. (defer). Élément SCRIPT.
dir : direction du texte faible/neutre. (ltr | rtl). Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FRAME, FRAMESET, IFRAME, PARAM, SCRIPT.
dir : directionnalité (obligatoire). (ltr | rtl). Élément BDO.
disabled : indisponible dans ce contexte. (disabled). Éléments BUTTON, INPUT, OPTGROUP, OPTION, SELECT, TEXTAREA.

E

enctype : type d'encodage. %ContentType;. Élément FORM.

F

face : liste de noms de police séparés par des virgules (déconseillé, DTD transitoire). CDATA. Éléments BASEFONT, FONT.
for : correspond à la valeur ID du champ. IDREF. Élément LABEL.
frame : quelles parties du contour restituer ? %TFrame;. Élément TABLE.
frameborder : bordures de cadre ou non ? (1 | 0). Éléments FRAME, IFRAME.

H

headers : liste des id des cellules de rubrique. IDREFS. Éléments TD, TH.
height : hauteur du cadre (déconseillé, DTD transitoire). %Length;. Élément IFRAME.
height : hauteur de la cellule (déconseillé, DTD transitoire). %Length;. Éléments TD, TH.
height : surclasser la hauteur. %Length;. Éléments IMG, OBJECT.
height : hauteur initiale (obligatoire) (déconseillé, DTD Transitoire). %Length;. Élément APPLET.
href : URI de la ressource reliée. %URI;. Éléments A, AREA, LINK.
href : URI qui fait office d'URI de base. %URI;. Élément BASE.
hreflang : code de langue. %LanguageCode;. Éléments A, LINK.
hspace : gouttière horizontale (déconseillé, DTD transitoire). %Pixels;. Éléments APPLET, IMG, OBJECT.
http–equiv : nom de l'en-tête de réponse HTTP. NAME. Élément META.

I

id : identifiant unique au document. ID. Tous les éléments sauf BASE, HEAD, HTML, META, SCRIPT, STYLE, TITLE.
ismap : utiliser une image cliquable côté serveur. (ismap). Éléments IMG, INPUT.

L

label : à utiliser dans les menus hiérarchiques. %Text;. Élément OPTION.
label : à utiliser dans les menus hiérarchiques (obligatoire). %Text;. Élément OPTGROUP
lang : code de langue. %LanguageCode;. Tous les éléments sauf APPLET, BASE, BASEFONT, BR, FRAME, FRAMESET, IFRAME, PARAM, SCRIPT.

language : nom de langage de script prédéfini (déconseillé, DTD transitoire). CDATA. Élément SCRIPT.
link : couleur des liens (déconseillé, DTD transitoire). %Color;. Élément BODY.
longdesc : lien vers une description longue (en complément de l'attribut alt). %URI;. Élément IMG.
longdesc : lien vers une description longue (en complément de l'attribut title) (DTD Frameset). %URI;. Éléments FRAME, IFRAME.

M

marginheight : marges verticales en pixels (DTD Frameset). %Pixels;. Éléments FRAME, IFRAME.
marginwidth : marges horizontales en pixels (DTD Frameset). %Pixels;. Éléments FRAME, IFRAME.
maxlength : nombre de caractères maxi pour les champs de texte. NUMBER. Élément INPUT.
media : prévu pour ces médias. %MediaDesc;. Élément STYLE.
media : pour restitution sur ces médias. %MediaDesc;. Élément LINK.
method : méthode HTTP utilisée pour soumettre le formulaire. (GET | POST). Élément FORM.
multiple : sélection simple par défaut. (multiple). Élément SELECT.

N

name : nom. CDATA. Éléments BUTTON, TEXTAREA.
name : permet aux applets de se trouver les uns les autres (déconseillé, DTD transitoire). CDATA. Élément APPLET.
name : nom du champ. CDATA. Élément SELECT.
name : nom du formulaire pour les scripts. CDATA. Élément FORM.
name : nom du cadre pour le ciblage. (DTD Frameset). CDATA. Éléments FRAME, IFRAME.
name : nom de l'image pour les scripts. CDATA. Élément IMG.
name : extrémité du lien nommée. CDATA. Élément A.
name : soumettre comme partie du formulaire. CDATA. Éléments INPUT, OBJECT.
name : pour appel par l'attribut usemap (obligatoire). CDATA. Élément MAP.
name : nom de propriété (obligatoire). CDATA. Élément PARAM.
name : nom des méta-informations (obligatoire). NAME. Élément META.
nohref : cette région est inactive. (nohref). Élément AREA.
noresize : autoriser l'utilisateur à redimensionner le cadre ? (DTD Frameset). (noresize). Élément FRAME.
noshade : (déconseillé, DTD transitoire). (noshade). Élément HR.

nowrap : suppression de la césure (déconseillé, DTD transitoire). (nowrap). Éléments TD, TH.

O

object : fichier d'applet sérialisé (déconseillé, DTD transitoire). CDATA. Élément APPLET.
onblur : l'élément a perdu l'attention. %Script;. Éléments A, AREA, BUTTON, INPUT, LABEL, SELECT, TEXTAREA.
onchange : la valeur de l'élément a changé. %Script;. Éléments INPUT, SELECT, TEXTAREA.
onclick : le bouton d'un pointeur a été cliqué. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
ondblclick : le bouton d'un pointeur a été double-cliqué. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
onfocus : l'élément a reçu l'attention. %Script;. Éléments A, AREA, BUTTON, INPUT, LABEL, SELECT, TEXTAREA.
onkeydown : une touche est appuyée. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
onkeypress : une touche a été pressée puis relâchée. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
onkeyup : une touche est relâchée. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
onload : tous les cadres ont été chargés (DTD Frameset). %Script;. Élément FRAMESET.
onload : le document a été chargé. %Script;. Élément BODY.
onmousedown : le bouton d'un pointeur a été appuyé. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
onmousemove : le bouton d'un pointeur a été déplacé à l'intérieur. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.

onmouseout : le bouton d'un pointeur a été déplacé en dehors. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
onmouseover : le bouton d'un pointeur a été déplacé sur. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
onmouseup : le bouton d'un pointeur a été relâché. %Script;. Tous les éléments sauf APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE.
onreset : le formulaire a été réinitialisé. %Script;. Élément FORM.
onselect : un texte a été sélectionné. %Script;. Éléments INPUT, TEXTAREA.
onsubmit : le formulaire a été soumis. %Script;. Élément FORM.
onunload : tous les cadres ont été retirés (DTD Frameset). %Script;. Élément FRAMESET.
onunload : le document a été retiré. %Script;. Élément BODY.

P

profile : dictionnaire de méta-informations nommé. %URI;. Élément HEAD.
prompt : message d'invite (déconseillé, DTD Transitoire). %Text;. Élément ISINDEX.

R

Readonly : lecture seule. (readonly). Élément TEXTAREA.
readonly : pour les boutons de type text et passwd. (readonly). Élément INPUT.
rel : types de liens progressifs. %LinkTypes;. Éléments A,LINK.
rev : types de liens régressifs. %LinkTypes;. Éléments A, LINK.
rows : liste de longueurs, par défaut : 100% (1 rangée) (DTD Frameset). %MultiLengths;. Élément FRAMESET.
rows : nombre de lignes (obligatoire). NUMBER. Élément TEXTAREA.
rowspan : nombre de rangées couvertes par la cellule. NUMBER. Éléments TD, TH.
rules : règles entre rangées et colonnes. %TRules;. Élément TABLE.

S

scheme : sélectionner une forme de contenu. CDATA. Élément META.
scope : portée des cellules de rubrique. %Scope;. Éléments TD, TH.
scrolling : barres de défilement ou non (DTD Frameset). (yes | no

| auto). Éléments FRAME, IFRAME.
selected : présélection. (selected). Élément OPTION.
shape : contrôle l'interprétation des coordonnées. %Shape;. Élément AREA.
shape : à employer avec les images cliquables côté client. %Shape;. Élément A.
size : taille (déconseillé, DTD Transitoire). %Pixels;. Élément HR.
size : taille de la police, [+|−]entier, par exemple size="+1", size="4" (déconseillé, DTD transitoire). CDATA. Élément FONT.
size : propre à chaque type de champ. CDATA. Élément INPUT.
size : taille de police de base pour les éléments FONT (obligatoire) (déconseillé, DTD transitoire). CDATA. Élément BASEFONT.
size : rangées visibles. NUMBER. Élément SELECT.
span : Les attributs de l'élément COL affectent « n » colonnes. NUMBER. Élément COL.
span : nombre de colonnes par défaut dans le groupe. NUMBER. Élément COLGROUP.
src : URI d'un script externe. %URI;. Élément SCRIPT.
src : pour les champs avec des images. %URI;. Élément INPUT.
src : source du contenu du cadre (DTD Frameset). %URI;. Éléments FRAME, IFRAME.
src : URI de l'image à incorporer (obligatoire). %URI;. Élément IMG.
standby : message à montrer pendant le chargement. %Text;. Élément OBJECT.
start : numéro commençant la séquence (déconseillé, DTD transitoire). NUMBER. Élément OL.
style : indications de style associées. %StyleSheet;. Tous les éléments sauf BASE, BASEFONT, HEAD, HTML, META, PARAM, SCRIPT, STYLE, TITLE.
summary : objet/structure pour sortie vocale. %Text;. Élément TABLE.

T

tabindex : position dans l'ordre de tabulation. NUMBER. Éléments A, AREA, BUTTON, INPUT, OBJECT, SELECT, TEXTAREA.
target : restituer dans ce cadre (DTD Transitoire). %FrameTarget;. Éléments A, AREA, BASE, FORM, LINK.
text : couleur du texte du document (déconseillé, DTD transitoire). %Color;. Élément BODY.
title : titre consultatif. %Text;. Tous les éléments sauf BASE, BASEFONT, HEAD, HTML, META, PARAM, SCRIPT, TITLE.
type : type de contenu consultatif. %ContentType;. Éléments A, LINK.

type : type de contenu pour l'attribut data. %ContentType;. Élément OBJECT.
type : type de contenu pour l'attribut value quand valuetype=ref. %ContentType;. Élément PARAM.
type : type de contenu du langage de script (obligatoire). %ContentType;. Élément SCRIPT.
type : type de contenu du langage de style (obligatoire). %ContentType;. Élément STYLE.
type : le genre de gadget voulu. %InputType;. Élément INPUT.
type : style de l'item de liste (déconseillé, DTD transitoire). %LIStyle;. Élément LI.
type : style de numérotation (déconseillé, DTD transitoire). %OLStyle;. Élément OL.
type : style de puces (déconseillé, DTD transitoire). %ULStyle;. Élément UL.
type : utiliser comme bouton de formulaire. (button | submit | reset). Élément BUTTON.

U

usemap : utiliser une image cliquable côté client. %URI;. Éléments IMG, INPUT, OBJECT.

V

valign : Alignement vertical dans les cellules. (top | middle | bottom | baseline). Éléments COL, COLGROUP, TBODY, TD, TFOOT, TH, THEAD, TR.
value : spécifier pour les boutons radio et les cases à cocher. CDATA. Élément INPUT.
value : par défaut, le contenu de l'élément. CDATA. Élément OPTION.
value : valeur de propriété. CDATA. Élément PARAM.
value : envoyé au serveur à la soumission. CDATA. Élément BUTTON.
value : réinitialise le numéro dans la séquence (déconseillé, DTD transitoire). NUMBER. Élément LI.
valuetype : comment interpréter la valeur ? (déconseillé, DTD transitoire). (DATA | REF | OBJECT). Élément PARAM.
version : une constante (déconseillé, DTD transitoire). CDATA. Élément HTML.
vlink : couleur des liens visités (déconseillé, DTD transitoire). %Color;. Élément BODY.
vspace : gouttière verticale (déconseillé, DTD transitoire). %Pixels;. Éléments APPLET, IMG, OBJECT.

W

`width` : largeur (déconseillé, DTD transitoire). `%Length;`. Élément `HR`.
`width` : largeur du cadre (DTD transitoire). `%Length;`. Élément `IFRAME`.
`width` : surclasser la largeur. `%Length;`. Éléments `IMG`, `OBJECT`.
`width` : largeur du tableau. `%Length;`. Élément `TABLE`.
`width` : largeur de la cellule (déconseillé, DTD transitoire). `%Length;`. Éléments `TD`, `TH`.
`width` : largeur initiale (obligatoire) (déconseillé, DTD transitoire). `%Length;`. Élément `APPLET`.
`width` : spécification de la largeur de colonne. `%MultiLength;`. Élément `COL`.
`width` : largeur par défaut des éléments `COL` contenus. `%MultiLength;`. Élément `COLGROUP`.
`width` : largeur (déconseillé, DTD transitoire). `NUMBER`. Élément `PRE`.

12.3. Compatibilité XHTML/HTML

Comme XHTML est une application XML, certaines habitudes parfaitement légales dans le HTML 4 fondé sur SGML doivent être amendées.

Rédaction correcte des documents

La rédaction correcte est un nouveau concept introduit par XML. Il signifie essentiellement qu'un élément doit toujours posséder une balise de fin ou être écrit selon une forme particulière (voir ci-dessous). En outre, tous les éléments doivent être imbriqués et non se chevaucher.

***Tableau 12.2 :** « La rédaction correcte »*

CORRECT	INCORRECT
éléments imbriqués	éléments se chevauchant
`<p>ceci est un paragraphe <em>avec emphase</em>.</p>`	`<p>ceci est un paragraphe <em>avec emphase.</p></em>`

Noms des éléments et des attributs en minuscules

Les documents XHTML doivent employer la casse « minuscules » pour tous les noms d'élément HTML et les noms d'attribut. Cette différence est nécessaire, car XML est sensible à la casse (`<li>` et `<LI>` sont des balises différentes).

Balises de fin obligatoires

Dans le HTML 4 fondé sur SGML, il est possible d'omettre la balise de fin de certains éléments, l'élément suivant créant une balise de fin implicite. Cette omission n'est plus autorisée dans le XHTML fondé sur XML. Tous les éléments autres que ceux déclarés dans la DTD comme `EMPTY` doivent posséder une balise de fin.

Tableau 12.3 : *XHTML : balise de fin obligatoire*

CORRECT	INCORRECT
éléments terminés	éléments non terminés
`<p>Ceci est un paragraphe.</p><p> Ceci est un autre paragraphe.</p>`	`<p>Ceci est un paragraphe.<p>Ceci est un autre paragraphe.`

Les éléments vides doivent toujours posséder une balise de fin ou la balise de début se terminer avec `/>`. Par exemple, `<br/>` ou `<hr></hr>`.

Tableau 12.4 : *Éléments vides*

CORRECT	INCORRECT
balises vides terminées	balises vides non terminées
` <hr/>`	` <hr>`

Vous devez placer un espacement avant le `/` et `>` de fin des éléments vides, par exemple `<br />`, `<hr />` et `<img src="karen.jpg" alt="Karen" />`. Utilisez également une syntaxe minimale pour les éléments vides, par exemple `<br />`, comme syntaxe alternative de `<br></br>` qui est autorisé par XML, car cela donne des résultats inattendus avec certains agents utilisateurs.

En cas toutefois d'occurrence vide d'un élément dont le modèle de contenu n'est pas EMPTY (par exemple, un titre ou un paragraphe vide), n'utilisez pas la forme minimisée : utilisez <p></p> et non <p />.

Encodage de caractères

Pour spécifier l'encodage de caractères dans le document, utilisez la spécification de l'attribut d'encodage dans la déclaration xml (par exemple <?xml version="1.0" encoding="EUC-JP"?>) et une déclaration meta http-equiv (par exemple <meta http-equiv= "Content-type" content='text/html; charset="EUC-JP"' />). La valeur de l'attribut d'encodage de l'instruction de traitement xml est prioritaire.

Remarques sur les éléments

Éléments script et style

En XHTML, les éléments script et style sont déclarés comme possédant un contenu de type #PCDATA. Ainsi, < et & seront-ils traités comme le début d'un balisage, et les entités comme < et & seront-elles reconnues comme des références d'entités par le processeur XML, soit respectivement < et &. Emballer le contenu des éléments script ou style à l'intérieur d'une section marquée CDATA évitera la transformation de ces entités.

```
<script>
 <![CDATA[
 ... contenu script non echappé ...
 ]]>
 </script>
```

Les sections CDATA sont reconnues par le processeur XML et apparaissent comme des nœuds dans le Modèle Objet de Document. Une alternative consiste à employer des scripts et des styles externes.

Utilisez des feuilles de style externes si votre feuille de style utilise < ou & ou]]> ou --. Utilisez des scripts externes si vos scripts utilisent < ou & ou]]> ou --. Les parseurs XML ont le droit d'éliminer le contenu des commentaires. Par conséquent, la pratique historique de « cacher » ses scripts et ses feuilles de style au sein d'un commentaire pour rendre les documents compatibles avec les anciens navigateurs n'est pas

conseillée : elle ne fonctionnera pas comme attendu dans les mises en œuvre basées sur XML.

Élément isindex

Ne mettez pas plus d'un élément `isindex` dans le `head` d'un document. L'élément `isindex` est abandonné en faveur de l'élément `input`.

Remarques sur les attributs

XML ne supporte pas la minimisation des attributs. La paire valeur/attribut doit être écrite au complet. Les noms d'attributs tels que `compact` et `checked` ne peuvent pas être pris comme éléments sans que leur valeur soit spécifiée.

Tableau 12.5 : *Valeur obligatoire*

CORRECT	INCORRECT
attributs non minimisés	attributs minimisés
`<dl compact="compact">`	`<dl compact>`

Quelques agents utilisateurs HTML sont toutefois incapables d'interpréter les attributs booléens quand ils apparaissent dans leur forme complète (non minimisée), tels que requis par XML 1.0. Notez que ce problème n'affecte pas les agents utilisateurs compatibles avec HTML 4. Cela concerne les attributs `compact`, `nowrap`, `ismap`, `declare`, `noshade`, `checked`, `disabled`, `readonly`, `multiple`, `selected`, `noresize`, `defer`.

Toutes les valeurs d'attributs doivent être mises entre guillemets, même celles qui semblent être numériques.

Tableau 12.6 : *Guillemets impératifs*

CORRECT	INCORRECT
valeurs d'attributs entre guillemets	valeurs d'attributs sans les guillemets
`<table rows="3">`	`<table rows=3>`

Dans les valeurs d'attributs, les agents utilisateurs ôteront les espacements de début et de fin des valeurs d'attributs et dresseront une séquence d'un ou plusieurs caractères d'espacement (tels que les retours de lignes) à un unique espacement « intermots » (un caractère d'espacement ASCII pour les écritures occidentales).

Évitez les retours de ligne et les caractères d'espacement multiples au sein des valeurs d'attributs. Ils seront traités illogiquement par les agents utilisateurs.

Quand une valeur d'attribut contient une éperluette, l'attribut doit être exprimé comme une référence d'entité du caractère (par exemple `&`). Par exemple, quand l'attribut `href` de l'élément `a` pointe vers un script CGI qui accepte des paramètres, il doit être exprimé comme ceci :

```
http://my.site.dom/cgi-bin/myscript.pl?class=guest&amp;name=user
```

plutôt que comme ça :

```
http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user.
```

Attributs id et name

HTML 4 a défini l'attribut `name` pour les éléments `a`, `applet`, `form`, `frame`, `iframe`, `img` et `map`. HTML 4 a également introduit l'attribut `id`. Ces deux attributs ont été conçus pour être utilisés comme des identificateurs partiels.

En XML, les identificateurs partiels sont de type `ID`, et il ne peut y avoir qu'un unique attribut `ID` par élément. En XHTML 1.0, l'attribut `id` est aussi défini de type `ID`. Pour s'assurer que les documents XHTML 1.0 sont des documents XML correctement structurés, les documents XHTML 1.0 *doivent* utiliser l'attribut `id` quand l'identificateur partiel est défini, même sur les éléments qui possèdent historiquement aussi un attribut `name`.

En XML, un URI qui termine par un identificateur partiel de la forme `#foo` ne se réfère pas aux éléments possédant un attribut `name="foo"`, mais aux éléments possédant un attribut défini de type `ID`, soit l'attribut `id` de HTML 4. Beaucoup de clients HTML existants ne maintiennent pas l'utilisation des attributs de type `ID` de cette manière, donc des valeurs identiques doivent être fournies pour les deux attributs afin d'assurer des compatibilités ascendante et descendante maximales (par exemple, `<a id="foo" name="foo">...</a>`).

Également, depuis que l’ensemble des valeurs légales définies pour les attributs de type `ID` est bien plus restreint que pour ceux de type `CDATA`, le type de l’attribut `name` a été changé en `NMTOKEN`. Cet attribut est contraint de manière à ce qu’il ne puisse pas avoir d’autres valeurs que celles de type `ID`, ou comme la production Name en XML 1.0 Section 2.5, production 5. Malheureusement, cette contrainte ne peut pas être exprimée dans les DTD XHTML 1.0. À cause de ces changements, la plus grande précaution est de rigueur lors de la conversion de vos documents HTML existants. Les valeurs de ces attributs doivent être uniques à l’intérieur d’un document, valides, et toute référence à ces identificateurs partiels (qu’ils soient internes ou externes) doit être mise à jour, même si les valeurs doivent être changées durant la conversion.

XHTML 1.0 a abandonné l’attribut `name` des éléments `a`, `applet`, `form`, `frame`, `iframe`, `img` et `map`, et il sera éliminé dans les versions suivantes.

Attributs lang et xml:lang

Utilisez les deux attributs `lang` et `xml:lang` lorsque vous spécifiez le langage d’un élément. La valeur de l’attribut `xml:lang` est prioritaire.

Exclusions SGML

SGML permet au rédacteur d’une DTD d’exclure la présence de certains éléments à l’intérieur d’un élément. Une telle interdiction (appelée « exclusion ») n’est pas possible en XML.

Par exemple, la DTD HTML 4 stricte interdit l’emboîtement d’un élément `a` dans un autre élément `a` à quelque profondeur que ce soit. Il n’est pas possible de définir ce type d’interdiction en XML. Même si cette interdiction ne peut pas être définie dans la DTD, certains éléments ne devraient pas être emboîtés.

Instructions de traitement

Vérifiez que les instructions de traitement s’exécutent sur les agents utilisateurs. Si la déclaration XML n’est pas incluse dans un document, le document peut uniquement employer les jeux de caractères par défaut UTF-8 ou UTF-16.

Modèle Objet de Document et XHTML

La Recommandation de Modèle Objet de Document niveau 1 définit les interfaces du Modèle Objet de Document pour XML et HTML 4. Le Modèle Objet de Document du HTML 4 spécifie que les noms des éléments et des attributs HTML sont retournés en casse « majuscules ». Le Modèle Objet de Document XML spécifie que les noms des éléments et des attributs sont retournés dans la casse spécifiée. En XHTML 1.0, les noms des éléments et des attributs sont spécifiés dans la casse « minuscules ». Cette différence apparente peut être fixée de deux manières :

- Les applications qui accèdent à des documents XHTML distribués avec le type de média Internet `text/html` *via* le DOM peuvent utiliser le DOM HTML et s'appuyer sur des noms d'éléments et d'attributs retournés en majuscules par ces interfaces.
- Les applications qui accèdent à des documents XHTML distribués avec le type de média Internet `text/html` ou `application/xml` peuvent également utiliser le DOM XML. Les noms des éléments et des attributs sont renvoyés en casse « minuscules ». Quelques éléments XHTML peuvent ou non apparaître dans l'arbre d'objets, car ils sont facultatifs dans le modèle de contenu (par exemple l'élément `tbody` à l'intérieur d'un tableau `table`). Cela se produit parce qu'en HTML 4 quelques éléments pouvaient être minimisés et leurs balises de début et de fin pouvaient être toutes les deux omises (une fonctionnalité SGML). Ce n'est pas possible en XML. Plutôt que de demander aux auteurs de documents d'insérer des éléments hors contexte, XHTML a rendu facultatifs les éléments. Les applications ont besoin de s'adapter en respectant cela.

Feuilles de style imbriquées (CSS) et XHTML

La Recommandation des feuilles de style imbriquées niveau 2 définit les propriétés appliquées à l'arbre d'analyse grammaticale du document HTML ou XML. Les différences dans l'analyse produiront différents résultats sonores ou visuels, selon les sélecteurs utilisés. Les indicateurs suivants réduisent cet effet pour des documents distribués sans modification des deux types de médias :

- Les feuilles de style CSS pour le XHTML doivent employer des noms d'éléments et d'attributs de casse « minuscules ».

- Dans les tableaux, l'élément `tbody` sera déduit par le parseur d'un agent utilisateur HTML, mais pas par le parseur d'un agent utilisateur XML. Par conséquent, vous devez toujours ajouter explicitement un élément `tbody` s'il se réfère à un sélecteur CSS.
- Au sein de l'espace de noms XHTML, les agents utilisateurs reconnaissent l'attribut `id` comme un attribut de type `ID`. Par conséquent, les feuilles de style doivent pouvoir continuer à employer la syntaxe raccourcie `#` du sélecteur même si l'agent utilisateur ne lit pas la DTD.
- Au sein de l'espace de noms XHTML, les agents utilisateurs reconnaissent l'attribut `class`. Par conséquent, les feuilles de style doivent pouvoir continuer à employer la syntaxe raccourcie « `.` » du sélecteur.
- Les CSS définissent des règles de conformité différentes pour les documents HTML et XML. Les règles HTML s'appliquent aux documents XHTML distribués en tant que HTML, les règles XML aux documents XHTML distribués comme XML.

DTD XHTML

Comme pour HTML, XHTML existe en trois « parfums » : strict, transitoire et jeux d'encadrement. Ces trois versions sont respectivement déclarées comme suit :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

La variante *frameset* reprend tous les balises et attributs de la variante *transitional*. Deux balises ont été rajoutée : `frameset` et `frame`.

Différences entre strict et transitional

L'analyse des différences permet de traduire facilement un document non conforme, ou conforme HTML 4.01 transitional, en document XHTML 1.0 strict. La variante strict oblige une séparation du contenu et de la présentation. Cela implique donc la suppression de tous les éléments et attributs de mise en forme, et l'obligation de passer par les CSS.

Le Tableau B.1 présente les éléments devenus obsolètes en XHTML strict et leurs alternatives.

Tableau 12.7 : *Éléments obsolètes en XHTML strict*

Élément	Alternative
frameset, frame	Utilisez la variante frameset du XHTML
iframe	Utilisez la variante transitional du XHTML
font, basefont	Utilisez les styles CSS `font` et ses dérivés : `font-family`, `font-size`, `font-weight`, etc.
u	Style `text-decoration : underline;`
s, strike	Style `text-decoration : line-through;`
menu, dir	Ces balises ont en fait la même fonction que `ul` ; seule l'apparence par défaut change. Remplacez-les par une liste `ul` et modifiez l'apparence avec les styles `list-style` et leurs dérivés (`list-style-type`, etc.), et éventuellement `margin` ou `padding`.
center	Le style `text-align : center;` à mettre sur l'élément parent pour centrer des éléments de type en ligne. Sur les éléments de type bloc, les styles suivants : `margin-left : auto; margin-right : auto;`
isindex	Élément `input`
applet	ÉÉlément `object`

Le Tableau suivant présente les attributs devenus obsolètes en XHTML strict et leurs alternatives.

Tableau 12.8 : *Attributs obsolètes en XHTML strict*

Attribut	Éléments concernés	Alternative
Align	div, p, h1 à h6, hr, object, img, input, legend, table, caption	Style `text-align` sur l'élément parent pour aligner des éléments de type en ligne. Sur des éléments de type bloc, les styles suivants : `margin-left`, `margin-right`.

Tableau 12.8 : Attributs obsolètes en XHTML strict		
Attribut	**Éléments concernés**	**Alternative**
`background`	`body`	Style `background-image`, qui peut être employé conjointement avec les styles `background-position` et `background-repeat` pour positionner et définir la répétition de l'image de fond. Il est possible d'utiliser ces styles sur pratiquement tous les éléments.
`bgcolor`	`body`, `table`, `tr`, `td`, `th`	Style `background-color`
`border`	`object`, `img`	Style `border`
`clear`	`Br`	Style `clear`
`compact`	`Ul`, `ol`, `dl`	Jouer sur les styles `margin` ou `padding`
`height`	`Td`, `th`	Style `height`
`hspace`, `vspace`	`object`, `img`	Style `margin`
`language`	`script`	Aucun. L'attribut `type` suffit : `script type="text/javascript"`.
`link`, `alink`, `vlink`	`body`	Définissez un style par défaut sur l'élément `a` ainsi que sur les pseudo-classes `:link` , `:visited` et `:active`.
`name`	`img`, `form`	Utilisez l'attribut `id`
`Noshade`	`hr`	Ensemble des styles `border` , en particulier `border-style`.
`Nowrap`	`th`, `td`	Style `white-space : nowrap;`
`Size`	`hr`	Style `height`
`start`	`Ol`	Utilisez les styles pour gérer des compteurs
`target`	`base`, `link`, `a`, `area`, `form`	Aucun

Tableau 12.8 : Attributs obsolètes en XHTML strict		
Attribut	**Éléments concernés**	**Alternative**
`text`	`body`	Style `color`
`type`	`ul, ol, li`	Style `list-style`
`value`	`li`	Utilisez les styles pour gérer des compteurs
`width`	`hr, pre, td, th`	Style `width`

Remarquez par ailleurs que, au sein des éléments `form`, `noscript` et `blockquote` , il ne peut plus y avoir d'éléments de type en ligne.

Pour convertir rapidement et de façon correcte un document HTML en document au format XHTML, servez-vous de HTML Tidy ou de l'éditeur/navigateur Amaya.

12.4. DTD et XML

XML a recours à un fichier nommé DTD (*Document Type Definition*, définition de document type) pour vérifier qu'un document XML respecte une syntaxe donnée. Une DTD constitue une grammaire qui permet de vérifier la conformité du document XML. La norme XML n'impose pas l'utilisation d'une DTD pour un document XML, mais elle impose en revanche le respect exact des règles de base de la norme XML.

La terminologie XML emploie les définitions suivantes :

- *Document bien formé* pour un document qui ne comporte pas de DTD mais respecte les exigences de la spécification.
- *Document valide* pour un document qui se conforme strictement aux règles d'une DTD.

Une DTD peut être définie de 2 façons :

- **De façon interne** : la grammaire est incluse au sein même du document.
- **De façon externe** : soit en appelant un fichier qui contient la grammaire à partir d'un fichier local, soit en y accédant par son URL.

C'est cette dernière solution qui est adoptée par les fichiers HTML : HTML 4 étant fondé sur XML, il repose sur des DTD. Vous l'avez vu lors de l'écriture de pages Web, puisque tout fichier HTML débute en principe par la spécification de la DTD employée :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

La préhistoire du développement de sites Web a fait croire à de nombreux concepteurs que le HTML pouvait s'écrire de manière assez libre, le navigateur se chargeant de rattraper les bêtises et d'afficher la page tant bien que mal.

C'est évidemment totalement faux. Cela n'était dû qu'au simple fait que les agents utilisateurs de l'époque étaient *non validants*, incapables de gérer correctement et totalement les spécifications des langages Web. C'est pourquoi un concepteur doit encore se battre pour faire fonctionner un site dans les principaux navigateurs.

Il est donc utile de s'intéresser un peu plus à la structure d'une DTD. Savoir lire une DTD est en effet presque obligatoire pour toute personne devant produire du code (XML, (X)HTML, etc.) valide. Une DTD constitue une référence beaucoup plus rapide à consulter que les longues spécifications publiées par le W3C. Elle indique clairement ce que le navigateur est en droit d'attendre.

Déclaration d'un élément

Pour pouvoir créer un document XML, il est utile dans un premier temps de définir les éléments qui peuvent être employés, et plus exactement les informations que vous souhaitez utiliser.

La définition d'un élément répond à la syntaxe suivante :

```
<!ELEMENT Nom Modèle >
```

Le paramètre `modèle` représente soit un type de données prédéfini, soit une règle d'utilisation de l'élément. Les types prédéfinis utilisables sont présentés dans le Tableau D.1.

Tableau 12.9 : *Types prédéfinis pour un élément*

Type prédéfini	Description
`ANY`	L'élément peut contenir tout type de données

Type prédéfini	Description
EMPTY	L'élément ne contient pas de données spécifiques
#CDATA	Le contenu n'est pas interprété par l'agent utilisateur
#PCDATA	L'élément doit contenir une chaîne de caractères. Ce contenu est interprété par l'agent utilisateur.

Tableau 12.9 : *Types prédéfinis pour un élément*

Autrement dit, un élément avec un contenu de type #PCDATA (*parsed character data*) verra ce contenu interprété par le navigateur, les balises éventuelles étant traitées comme telles, tandis qu'avec #CDATA le contenu est considéré comme une simple chaîne de caractères...

Ainsi un élément nommé titre contenant un type #PCDATA sera déclaré comme suit dans la DTD :

```
<!ELEMENT titre (#PCDATA) >
```



Ecriture de #PCDATA

Le mot-clé #PCDATA doit impérativement figurer entre parenthèses. Vous obtiendrez sinon une erreur du parseur.

Cet élément pourra être écrit de la façon suivante dans le document XML :

```
<titre>Bible C++</titre>
```

Il est également possible de définir des règles d'utilisation : les éléments XML qu'un élément peut ou doit contenir. Cette syntaxe se fait à l'aide de notations spécifiques, présentées dans le Tableau D.2 :

Tableau 12.10 : *Opérateurs des règles d'utilisation d'un élément*

<table>
<tr><th>Opérateur</th><th>Exemple</th><th>Signification</th></tr>
<tr><td>+</td><td>a+</td><td>L'élément a doit être présent au minimum une fois</td></tr>
<tr><td>*</td><td>a*</td><td>L'élément a peut être présent plusieurs fois (ou aucune)</td></tr>
<tr><td>?</td><td>a?</td><td>L'élément a est facultatif.</td></tr>
<tr><td>|</td><td>a|b</td><td>Les éléments a ou b peuvent être présents</td></tr>
</table>

Tableau 12.10 : Opérateurs des règles d'utilisation d'un élément		
Opérateur	**Exemple**	**Signification**
,	a,b	L'élément a doit être présent et suivi de l'élément b
()	(a,b)+	Regroupement d'éléments en vue de leur appliquer les autres opérateurs. Ici, la présence d'un élément a suivi d'un élément b est obligatoire.

Vous pouvez ainsi créer la déclaration suivante dans une DTD :

```
<!ELEMENT livre (titre,editeur,auteur,date),url? >
<!ELEMENT titre (#PCDATA) >
<!ELEMENT editeur (#PCDATA) >
<!ELEMENT auteur (#PCDATA) >
<!ELEMENT url (#PCDATA) >
```

Cette déclaration peut aboutir à un document XML du style :

```
<livre>
   <titre>La Bible C++</titre>
   <editeur>Micro Application </editeur>
   <auteur>Horstamn C, Budd T. </auteur>
   <date>2004</date>
   <url>www.microapp.com </url>
</livre>
```

ou bien :

```
<livre>
   <titre>La Bible C++</titre>
   <editeur>Micro Application </editeur>
   <auteur>Horstamn C, Budd T. </auteur>
   <date>2004</date>
</livre>
```

Déclaration d'attributs

Vous pouvez ajouter des propriétés à un élément particulier en lui affectant un attribut, c'est-à-dire une paire clé/valeur. Avec XML, la syntaxe de définition d'un attribut est la suivante :

```
<! ATTLIST élément attribut type >
```

`type` représente le type de donnée de l'attribut. Il peut être :

- Littéral : vous affectez une chaîne de caractères à un attribut. Vous le déclarez ainsi à l'aide du mot-clé `CDATA`.

- Énumération : permet de définir une liste de valeurs possibles pour un attribut donné, afin de limiter le choix de l'utilisateur. La syntaxe de ce type d'attribut est :

```
<! ATTLIST élément attribut (valeur1 | valeur2 | ... ) >
```

- Pour définir une valeur par défaut, faites suivre l'énumération par la valeur désirée placée entre guillemets :

```
<! ATTLIST élément attribut (valeur1 | valeur2 )
"valeur_par_défaut" >
```

- Atomique : permet de définir un identifiant unique pour chaque élément grâce au mot-clé `ID`.

Chacun de ces types d'attributs peut être suivi d'un mot-clé particulier qui permet de spécifier le niveau de nécessité de l'attribut :

- `#IMPLIED` : l'attribut est facultatif.
- `#REQUIRED` : l'attribut est obligatoire.
- `#FIXED` : l'attribut possède une valeur par défaut s'il n'est pas défini. Il doit être immédiatement suivi de la valeur placée entre guillemets.

Une déclaration d'attribut peut ainsi se présenter comme ceci :

```
<! ATTLIST livre IDlivre ID #REQUIRED
  niveau(debutant|intermediaire|expert)"expert" >
```

L'élément livre possède deux attributs `IDlivre` et `niveau`. Le premier attribut est de type atomique : c'est un identifiant unique obligatoire. L'attribut `niveau` peut être débutant, intermédiaire ou expert, cette dernière valeur étant affectée par défaut.

Déclaration de notations

XML permet la définition d'une application par défaut à lancer pour ouvrir des documents non XML encapsulés dans le code XML. Il est ainsi possible d'associer les images JPG au programme Paint Shop Pro (psp.exe) grâce à la syntaxe suivante :

```
<! NOTATION jpg SYSTEM "psp.exe">
```

Déclaration d'entités

XML permet de créer des entités, c'est-à-dire de déclarer un groupe d'éléments sous un nom afin de ne pas avoir à réécrire ces derniers plusieurs fois dans la DTD s'ils se répètent, dans le même esprit que les macros dans les éditeurs de texte. Le recours à des entités dans un document XML permet une meilleure lisibilité, un meilleur contrôle du contenu et améliore la facilité de mise à jour.

XML possède plusieurs types d'entités : les entités générales, les entités paramètres et les entités caractères.

Entités générales

Les entités générales servent à définir des éléments pouvant être substitués dans le corps du document XML (même s'ils sont définis au sein de la DTD et non du document XML lui-même). La syntaxe d'une entité générale est la suivante :

```
<!ENTITY nom_entite "Contenu de l'entite">
```

Il est par exemple possible de déclarer l'entité générale suivante dans la DTD :

```
<!ENTITY site "microapp.com">
```

Les entités définies dans la DTD peuvent ainsi être utilisées dans le code XML en les appelant avec la syntaxe suivante :

```
&nom__entite;
```

La déclaration précédente pourra donc donner un document XML du style :

```
<livre>
   <titre>La Bible C++</titre>
   <editeur>Micro Application </editeur>
   <auteur>Horstamn C, Budd T. </auteur>
   <date>2004</date>
   <url>www.@&site;</url>
</livre>
```

Le parseur transformera automatiquement chacune des entités contenues dans le code XML en chaîne de caractères :

```
<livre>
   <titre>La Bible C++</titre>
   <editeur>Micro Application </editeur>
   <auteur>Horstamn C, Budd T. </auteur>
```

```
   <date>2004</date>
   <url>www.microapp.com</url>
</livre>
```

Les entités générales peuvent aussi être contenues dans un fichier extérieur (afin d'être utilisées par plusieurs DTD, par exemple). Elles portent alors le nom d'*entités externes*. La syntaxe d'une entité externe est la suivante :

```
<!ENTITY nom_entite SYSTEM "nom_fichier">
```

Dans l'exemple suivant, le contenu du fichier *niv.txt* (situé dans le même répertoire que la DTD) sera inséré dans le fichier XML à chaque fois que l'entité `∋` sera rencontrée :

```
<!ENTITY niv SYSTEM "niv.txt">
```

Entités paramètres

Une entité paramètre permet d'employer des entités dans les DTD elles-mêmes. La syntaxe est la suivante :

```
<!ENTITY % nom_entite definition>
```

Le nom de l'entité doit toutefois respecter certaines règles :

- Commencer par une lettre ou un trait de soulignement (*underscore* _).
- Être composé uniquement de lettres, de chiffres, de tirets (–), de traits de soulignement (_), de points (`.`) ou du caractère deux points (`:`).

Voici un exemple de déclaration d'entité paramètre :

```
<!ENTITY % niv #REQUIRED
   niveau(debutant|intermediaire|expert) "expert">
```

Il est également possible (comme pour les entités générales) de définir une entité paramètre externe grâce à la syntaxe suivante :

```
<!ENTITY % nom_entite SYSTEM "fichier.dtd">
```

La DTD `fichier.dtd` sera un fichier comportant la déclaration d'entité paramètre :

```
<!ENTITY % nom_entite "definition">
```

Entités caractères

Les entités caractères sont des caractères réservés du XML représentés sous forme d'entités générales afin de pouvoir insérer ces caractères réservés dans le document XML. Les principales entités caractères sont présentées dans le Tableau D.3.

Tableau 12.11 : *Principales entités caractère*

Entité caractère	Représentation
&	&
<	<
>	>
'	'
"	"

Il est également possible de définir des entités caractères pour n'importe quel caractère en utilisant le code hexadécimal du caractère :

```
<!ENTITY nom_entite "&#xCODEHEXA;">
```

Par exemple :

```
<!ENTITY ccedille "&#x00E7;">
```

Même si cette annexe peut sembler succincte, elle devrait vous permettre de mieux comprendre les DTD des spécifications HTML 4.01 et XHTML 1.

12.5. Sélecteurs CSS

La version 3 des sélecteurs CSS (**http://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html**) propose des sélecteurs nombreux et très sophistiqués, capables de répondre désormais à pratiquement toutes les situations. Les sélecteurs sont classés dans le tableau suivant par ordre d'apparition dans les versions CSS, afin de montrer l'évolution.

Tableau 12.12 : *Evolution des sélecteurs CSS de la version 1 à la version 3*		
Motif	**Signification**	**CCS**
`E`	Un élément de type E	1
`E:link/visited`	Un élément E d'une ancre hypertexte dont la cible n'a pas encore été visitée (`:link`) ou l'a été (`:visited`)	1
`E::first-line`	Le première ligne mise en forme d'un élément E	1
`E::first-letter`	La première letttre mise en forme d'un élément E	1
`E.warning`	Un élément E dont la classe est « *warning* ». Le langage du document spécifie le mode de détermination de la classe.	1
`E#monid`	Un élément E dont l'ID est « *monid* ».	1
`E F`	Un élément F descendant d'un élément E	1
`E:active/hover/focus`	Un élément E pendant certaines actions de l'utilisateur	1 et 2
`*`	N'importe quel élément	2
`E[attribut]`	Un élément E avec un attribut « *attribut* »	2
`E[attribut="valeur"]`	Un élément E avec un attribut « *attribut* » dont la valeur est « *valeur* »	2
`E[attribut~="valeur"]`	Un élément E avec un attribut « *attribut* » dont la valeur est une liste de valeurs séparées par des virgules dont l'une est égale à « *valeur* »	2
`E:first-child`	Un élément E, premier enfant de son parent	2
`E:lang(fr)`	Un élément E en langue « `fr` ». Le langage du document spécifie le mode de détermination de la langue	2
`E::before`	Contenu généré avant un élément E	2
`E::after`	Contenu généré après un élément E	2
`E > F`	Un élément F enfant d'un élément E	2
`E + F`	Un élément F immédiatement précédé par un élément E	2
`E[attribut^="valeur"]`	Un élément E avec un attribut « *attribut* » dont la valeur commence exactement par la chaîne « *valeur* »	3

Tableau 12.12 : *Evolution des sélecteurs CSS de la version 1 à la version 3*		
Motif	**Signification**	**CCS**
`E[attribut$="valeur"]`	Un élément E dont la valeur de l'attribut « *attribut* » se termine exactement par la chaîne « *valeur* »	3
`E[attribut*="valeur"]`	Un élément E dont la valeur de l'attribut « *attribut* » contient la sous-chaîne « *valeur* »	3
`E[hreflang|="en"]`	Un élément E dont l'attribut « `hreflang` » contient une liste séparée par des tirets qui commence (à gauche) par « `en` »	3
`E:root`	Un élément E racine du document	3
`E:nth-child(n)`	Un élément E, $n^{ième}$ enfant de son parent	3
`E:nth-last-child(n)`	Un élément E, $n^{ième}$ enfant de son parent en partant du dernier	3
`E:nth-of-type(n)`	Un élément E, $n^{ième}$ descendant de ce type	3
`E:nth-last-of-type(n)`	Un élément E, $n^{ième}$ descendant de ce type à partir du dernier	3
`E:last-child`	Un élément E, $n^{ième}$ dernier enfant de son parent	3
`E:first-of-type`	Un élément E, premier descendant de ce type	3
`E:last-of-type`	Un élément E, dernier descendant de ce type	3
`E:only-child`	Un élément E, seul enfant de son parent	3
`E:only-of-type`	Un élément E, seul descendant de ce type	3
`E:empty`	Un élément E dépourvu d'enfant (y compris tout nœud texte)	3
`E:target`	Un élément E cible de l'URI	3
`E:enabled/disabled`	Un élément d'interface utilisateur E activé ou désactivé	3
`E:checked`	Un élément d'interface utilisateur E coché (bouton d'option ou case à cocher)	3
`E::selection`	La portion d'un élément E actuellement sélectionnée/mise en surbrillance par l'utilisateur	3
`E:not(s)`	Un élément E qui ne correspond pas au sélecteur simple *s*	3
`E ~ F`	Un élément F précédé par un élément E	3

Comme le montre ce tableau, les possibilités d'enrichissement d'un contenu à l'aide d'une feuille de style sont désormais extrêmement riches. CSS permet en outre de combiner à l'aide d'opérateurs les sélecteurs simples, ce qui élargit encore le champ des possibilités.

12.6. Ressources Web

De nombreuses autres ressources sont présentes sur le Web : il est fortement recommandé d'y accéder, car les ressources en ligne sont actualisées plus fréquemment que ne peut l'être cet ouvrage.

Spécifications

Spécification HTML 4.01 : www.w3.org/TR/html401 (anglais), www.la-grange.net/w3c/html4.01/cover.html (français).
Spécification du Modèle Objet de Document (DOM) : www.w3.org/TR/REC-DOM-Level-1/
Recommandation XHTML 1.0 : www.w3.org/TR/2002/REC-xhtml1-20020801/ (anglais), www.la-grange.net/w3c/xhtml1/ (français).
Site Web Unicode : www.unicode.org/

Outils de création Web

Logiciel commerciaux

WebExpert : http://softwares.visicommedia.com/fr/products/webexpert/overview.html
Adobe Golive : www.adobe.fr/products/golive/main.html
Namo Web Editor : www.namo.com/products/webeditor.php
HotDog Professional 7.0.1 : www.sausage.com/
WebFast : www.internetspirit.com/

Logiciels gratuits

1st Page 2000 : www.evrsoft.com/1stpage2.shtml
Amaya : www.w3.org/Amaya/
Mozilla Composer : www.archilinux.org/mozilla/composer.html
Téléchargement de la suite Mozilla : www.mozilla-europe.org/fr/
Netscape Composer : http://wp.netscape.com/communicator/composer/v4.0/

Nvu : http://frenchmozilla.sourceforge.net/nvu/ Le site principal anglais est www.nvu.com Un tutoriel très complet couvrant l'installation, la configuration et l'utilisation de Nvu (fondé sur la version 0.9) est également proposé sur le site de Framasoft (www.framasoft.net/article2656.html).
SoThink HTML Editor (CutePage) : www.sothink.com/htmleditor/

Navigateurs

Internet Explorer : www.microsoft.com/france/internet/produits/ie/default.mspx
Netscape Navigator : www.telechargement.netscape.fr/telechargement/netscape7/
Mozilla : www.mozilla-europe.org/fr/
Firefox : www.mozilla-europe.org/fr/products/Firefox/
Un excellent comparatif entre Internet Explorer et Firefox est proposé à l'adresse http://emmanuel.clement.free.fr/navigateurs/comparatif.htm
Opera : www.opera.com/
Camino : www.mozilla-europe.org/fr/
Lynx : un excellent tutoriel en français peut être trouvé à l'adresse http://dominique.guebey.club.fr/tekno/lynx/index.htm La page officielle est http://lynx.browser.org/
pwWebSpeak : une version de démonstration de ce navigateur texte-parole peut être trouvée à l'adresse www.prodworks.com

Images

Fonds d'écran : www.atelier-duotang.com/tuiles/modeles/fonds.php?page=1
Ulead SmartSaver : www.ulead.com/webutilities/frwhere.htm
Paint Shop Pro (129 $): www.jasc.com/products/paintshoppro/
The Gimp : http://gimp-win.sourceforge.net/
SRGB : un espace de couleur par défaut standard pour Internet, version 1.10, M. Stokes, M. Anderson, S. Chandrasekar et R. Motta, 5 novembre 1996. www.w3.org/Graphics/Color/sRGB

Sites de gifs animés

dgif.legif.com/
www.yatoula.com/
www.icone-gif.com/
www.chez.com/gif/

www.01gif.com/
gif.webgratuit.com/

Éditeurs d'images animées

Microsoft Gif Animator (gratuit) : www.zdnet.fr/telecharger/windows/fiche/0,39021313,11010272s,00.htm
A Smaller GIF (28 $) : www.peda.com/smaller/download.html
Gif.gIf.giF (28 $) : www.peda.com/ggg/download.html
Ulead GIF Animator 5 (49 $) : www.ulead.com
Animated Shop/Paint Shop Pro (29/129 $) : www.jasc.com/products/paintshoppro/

Feuilles de style

Spécification CSS2 : www.w3.org/TR/REC-CSS2/cover.html#minitoc
Propriétés des feuilles de style CSS2 : www.w3.org/TR/REC-CSS2/propidx.html
Spécifications sur les règles d'héritage CSS2 : www.w3.org/TR/REC-CSS2/cascade.html
Le W3C a créé plusieurs styles basiques à utiliser sur une page Web. Visitez www.w3.org/StyleSheets/Core/
Vous trouverez l'ensemble des propriétés des feuilles de style auditives à l'adresse www.w3.org/TR/REC-CSS2/aural.html

Scripts

JavaScript

Pour commencer, visitez www.hotwired.com/webmonkey/javascript/
La JavaScript Source d'Internet World se situe à l'adresse http://javascript.internet.com/
HotSyte (www.serve.com/hotsyte/) propose des ressources JavaScript, offrant de nombreux programmes téléchargeables.
Il existe des dizaines de sites consacrés à des exemples de code JavaScript, parmi lesquels www.toutjavascript.com, javascript.internet.com/, javascriptkit.com/, www.javascriptfr.com/ et bien d'autres.
Le groupe de discussion *Usenet comp.lang.javascript* est un bon endroit où poser des questions et obtenir les informations les plus fraîches sur JavaScript.

Une foire aux questions (FAQ) JavaScript peut être trouvée à l'adresse http://idm.internet.com/faq/js-faq.shtml
Une excellente ressource JavaScript est située à l'adresse www.webreference.com/javascript

JScript

La page officielle du créateur Microsoft est http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jslrfjscriptlanguagereference.asp
Vous trouverez d'autres exemples aux adresses www.asp-fr.net/jscript/ et www.pageresource.com/jscript/ (entre autres).
Vous pouvez trouver des milliers de scripts JavaScript à l'adresse www.javascripts.com

ECMA-Script

www.ecma-international.org/publications/standards/Ecma-262.htm

VBScript

La page officielle du créateur Microsoft se trouve à l'adresse http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vtorivbscript.asp
Vous trouverez d'autres exemples et tutoriels aux adresses www.intranetjournal.com/corner/wrox/progref/vbt/, www.asp-fr.net/vbscript/, www.asp-php.net/tutorial/scripting/index.php (entre autres).

Java

De nombreuses ressources de script sont disponibles à l'adresse www.irt.org/ Ce site propose d'ailleurs des ressources sur pratiquement tous les sujets abordés dans ce livre.
De nombreuses autres ressources Java, parmi lesquelles des applets, peuvent être trouvées à l'adresse www.developer.com/java/

Cookies

La foire aux questions (FAQ) Unofficial Cookie se trouve à l'adresse www.cookiecentral.com/unofficial_cookie_faq.htm

Modules complémentaires

Shockwave : www.macromedia.com/shockwave/download/download.cgi?P1_Prod_Version=ShockwaveFlash
Adobe Acrobat Reader : www.adobe.fr/products/acrobat/readstep2.html
RealPlayer : http://france.real.com/player/?&src=ZG.fr.idx,ZG.fr.rp.rp.hd.def
Ressources MIDI : www.midi.com
QuickTime : www.apple.com/quicktime/download/win.html

Logiciels FTP

CuteFTP : www.cuteftp.com/cuteftp/
LeechFTP : http://stud.fh-heilbronn.de/~jdebis/leechftp/downloads.html. Ce produit, dont il existe une version française, va prochainement être remplacé par BitBeamer, du même auteur.
WS- FTP 95 : nombreux sites, dont www.sharewarejunkies.com/8zwd7/ws_ftp_95_le.htm
FileZilla : nombreux sites, dont le site officiel http://sourceforge.net/project/showfiles.php?group_id=21558.

Vous pourrez télécharger de nombreux autres clients FTP gratuits à partir des adresses suivantes :
www.01net.com/telecharger/windows/Internet/ftp/
www.tutoriels.com/downloads.php?id=57
www.jetelecharge.com/softs-49-1.php
http://telecharger.presence-pc.com/categorie.asp?num=43&page=1
et bien d'autres encore…

Chapitre 13

Glossaire

A

Accessibilité : Procurer un accès universel à un site Web à l'aide de n'importe quel dispositif entrée-sortie comme : reconnaissance vocale, lecteurs d'écran ou interpréteurs braille, dispositifs de pointage non standard et TTY.

ActiveX : Technique propriétaire Microsoft permettant au navigateur Web d'incorporer et de contrôler des applications dans une page Web.

Adresse IP : Numéro unique utilisé par les ordinateurs afin de s'identifier sur Internet. Les utilisateurs se connectant par modem téléphonique possèdent souvent des adresses IP dynamiques, afin de minimiser le nombre d'IP disponibles.

Alignement : Positionnement du contenu d'une page par rapport aux marges : gauche, droite ou centré.

Animation : Graphisme à multiples images, un fichier Shockwave ou une applet d'animation Java semblant s'animer à l'écran.

API (*Application Programming Interface*) : Dans un serveur Web, un environnement de programmation permettant la rédaction de programmes intégrés pour des scripts côté serveur ou autres tâches du serveur.

Applet : Miniprogramme, généralement écrit en Java, qui exécute une application contenue dans une page Web.

Application : Programme indépendant qui possède un but particulier. Un programme JavaScript, une applet Java ou même un ensemble de pages Web peuvent constituer une application.

Architecture : Combinaison d'un microprocesseur et du matériel identifiant un modèle d'ordinateur (x86, SPARC, Macintosh, etc.).

Archive : Un ou plusieurs fichiers renfermant d'autres fichiers dans un but de sauvegarde ou de référence ultérieure.

ASCII : Données en texte pur, ni codées ni compressées.

Aspect : Combinaison des couleurs de document, des images et des outils de navigation conférant à un site une image homogène.

Assistant : Petit programme, le plus souvent incorporé à un autre programme, automatisant une tâche complexe ou guidant à travers les étapes successives d'un concept peu familier.

Attribut : Influe sur un élément HTML. Un attribut détermine les propriétés d'un élément.

Attribut (déclaration d') : En XML, une déclaration d'attribut définit la façon dont les attributs peuvent être utilisés et dans quels éléments.

Auditive (feuille de style) : Feuille de style en cascade (CSS) contenant des propriétés spécifiques aux navigateurs Web audio.

Autorisation : Permission de lire, d'écrire ou d'exécuter des fichiers accordée à des individus, des groupes ou des utilisateurs anonymes. Voir aussi **droit d'accès**.

Authentification de base : Nom d'utilisateur et/ou mot de passe exigés pour pouvoir accéder à un fichier ou à un répertoire, sans que la transaction (mot de passe compris) soit forcément cryptée.

B

`background` : Utilisé comme attribut pour l'élément `BODY`, il spécifie l'image d'arrière-plan à utiliser. En tant que propriété de feuille de style, il détermine la couleur d'arrière-plan de l'élément défini.

Balise : Chaîne de caractères débutant par < et finissant par >, qui ouvre ou ferme un élément HTML.

Balise (éditeur de type) : Éditeur affichant à l'écran les balises HTML alors que vous éditez votre page et possédant des raccourcis pour faciliter l'insertion des différentes balises.

Bande passante : Taux maximal de transfert d'un lien.

Barre d'outils : Ensemble de boutons et d'icônes d'un programme donnant accès aux outils et fonctionnalités de celui-ci.

Base de données : Un ou plusieurs fichiers rassemblant des données individuelles, telles que les noms d'un carnet d'adresses ou les produits d'un catalogue.

Barre de navigation : Partie d'une page Web donnant accès aux différentes pages et/ou actions possibles sur un site Web.

Bien formé : Terme appliqué à un document respectant la spécification à laquelle il fait référence (XML surtout).

Binaire : Données codées ou compressées, telles que des images, des archives, des programmes ou tout fichier non ASCII.

Bloc (élément) : Élément HTML qui insère un saut de ligne automatique avant sa balise d'ouverture et après sa balise de fermeture.

`BODY` : Corps du document : partie qui renferme le contenu visible d'une page.

Boîte de dialogue : Fenêtre indépendante qui apparaît pour poser une question, ou servir d'alerte ou d'avertissement à l'utilisateur. Une boîte de dialogue exige en général une réponse avant d'être fermée.

Bordure : Ligne ou frontière autour d'un élément tel qu'un tableau ou une image.

Bouton (aspect) : Transformer une image en lui donnant l'apparence d'un bouton « cliquable ».

C

Cache : Emplacement du disque dur où le navigateur stocke les fichiers temporaires.

Cadre (*frame*) : Panneau individuel dans un jeu d'encadrement (*frameset*) permettant de voir une seule page Web à la fois.

Caractères (jeu de) : Un jeu de caractères (*charset*) est l'ensemble des caractères et symboles utilisés dans un document. Par exemple, l'alphabet japonais utilise un jeu différent de celui de l'alphabet de l'Europe de l'Ouest.

Casse (sensibilité à la) : Différence entre majuscules et minuscules.

Cellule : Case individuelle d'un tableau.

CGI (*Common Gateway Interface*) : Interface de programmation qui permet à un utilisateur Web anonyme d'exécuter un programme (tel un script Perl) sur le serveur.

Champ (de formulaire) : Élément individuel qui récupère une saisie de l'utilisateur.

Chiffrage : Fait de crypter les données afin qu'elles ne puissent être lues sans l'application d'un algorithme de déchiffrage précis.

Classe : Sélecteur de feuille de style permettant d'appliquer une définition de style à n'importe quel élément en utilisant l'attribut facultatif `class`.

Client : Partie utilisateur d'une connexion réseau. L'ordinateur ou le programme client ne gère que quelques connexions simultanément, toutes les connexions étant initiées par lui-même.

Clip art : Image gratuite ou peu coûteuse pouvant être intégrée à tout document. Vérifiez toujours les droits d'auteurs et contraintes d'utilisation avant d'utiliser un quelconque clip art.

Code : Instructions sous-jacentes destinées à l'ordinateur. Le code source HTML n'est pas compilé et peut facilement être consulté.

Codec : Code sous-jacent qui permet la lecture d'un fichier d'un format précis. Essentiellement employé pour les fichiers son, image et vidéo. La majorité des codecs usuels sont intégrés aux logiciels concernés, mais il peut être à l'occasion nécessaire de télécharger un codec complémentaire.

Colonne : Affichage vertical de données dans un tableau.

Commentaire : Chaîne de caractères de type texte non affichée par le navigateur Web.

Compatibilité inter navigateurs : Rendre un site Web compatible avec la plupart, sinon tous les navigateurs Web, en général en n'utilisant que les standards Web.

Compteur : Script ou SSI (*Server Side Include*) comptant et affichant le nombre de touches reçues par la page active.

Compression : Utilisation d'un algorithme pour réduire la taille d'un fichier par élimination des données redondantes. Largement employé pour les fichiers image.

Concepteur : Professionnel du Web qui organise, rédige ou met en œuvre un site Web.

Contenu : Texte, images et fichiers multimédias placés dans une page Web.

Contenu (notation du) : Pratique consistant à décrire tout contenu potentiellement agressif ou dérangeant dans un élément `META` ou un commentaire afin de procurer un contrôle parental sur l'expérience Web des enfants.

Contextuel (élément) : Élément procurant des informations quant à la signification de l'élément marqué, et non uniquement sur la façon dont il doit être affiché.

Cookie : Petit fichier qui stocke de petits bouts d'informations afin d'aider les concepteurs de sites à les rendre plus interactifs ou pour permettre une personnalisation du site.

Couleurs de document : Couleurs du fond, du texte et des liens hypertextes utilisés dans un document, tels qu'ils sont définis dans un élément (souvent `BODY`) ou dans une feuille de style.

Courriel : Courrier électronique. Message envoyé rapidement et simplement sur Internet, à l'aide du protocole SMTP (*Standard Mail Transfer Protocol*), grâce à un logiciel de messagerie électronique.

D

Défilement : Utilisation des touches de direction du clavier ou des barres de défilement pour afficher les informations hors écran en raison des limites imposées par la taille de l'écran, mais appartenant au document courant.

Définition (liste de) : Liste HTML possédant deux types de membres : le terme et sa définition.

Descripteur de média : Dans une feuille de style, le type de l'agent utilisateur qui va afficher la page Web.

Document Type Declaration (DOCTYPE) : Le DTD est un document définissant la façon dont les éléments HTML doivent être affichés, ainsi que les attributs correspondant à ces éléments.

DOM (*Document Object Model*) : Le Modèle Objet de Document est un moyen de nommer les objets d'un document afin de pouvoir les utiliser dans un script.

Donnée : Information, généralement transmise ou stockée dans un ordinateur.

Droit d'accès : Permission de lire, d'écrire ou d'exécuter des fichiers, accordée à des individus, des groupes ou des utilisateurs anonymes.

Dynamique (HTML) ou **DHTML** : Rend HTML dynamique et modifiable à l'aide de langages de script et du DOM. Désormais déconseillé.

E

ECMAScript : Tentative de normalisation du langage de script employé avec HTML. Correspond à un « tronc commun » entre JavaScript et JScript.

Elément : Conteneur HTML qui possède une signification particulière pour un document.

Elément (déclaration) : En XML, déclarer un élément et son utilisation dans le DTD.

Email : Voir **courriel**.

En ligne : Fait d'être connecté à Internet. En HTML, caractérise également un élément qui s'insère à la suite du précédent et avant le suivant sans insertion de saut de ligne.

En-tête HTTP : Information envoyée automatiquement avec un document Web lors du transfert HTTP. Il peut contenir le nom du fichier, sa date, sa taille, ainsi que toute information mentionnée dans des éléments `META`.

Enfant (élément) : Tout élément inclus dans un autre élément HTML.

Entité (déclaration) : En XML, déclarer un objet comme devant être remplacé par un fichier externe ou par une donnée fréquemment utilisée, comme une adresse électronique ou des informations de droits d'auteur.

Espace vierge : Espace inutilisé autour du texte ou d'une image dans un programme de PAO ou la publication Internet. En HTML, certains caractères sont interprétés, comme les retour-chariot, les tabulations et les espaces excédentaires.

Exposant : Texte « flottant » au-dessus de la ligne normale de texte.

F

FAI : Fournisseur d'accès à Internet. Correspond à l'anglo-saxon ISP (*Internet Service Provider*).

FAQ (Foire aux questions ou *Frequently Asked Questions*) : Un fichier d'aide se présentant sous la forme d'une suite de questions/réponses.

Favoris : Ensemble de fichiers utilisés par Internet Explorer pour enregistrer les adresses des sites fréquemment visités. Correspond aux signets Netscape/Firefox.

Feuille de style : Fichier ou partie de document Web décrivant la façon d'afficher des éléments HTML individuels dans un navigateur Web.

Firefox : La nouvelle génération du navigateur Web Mozilla.

Formulaire : En HTML, un élément permettant aux utilisateurs de saisir des données traitées par la suite *via* un script côté client ou un programme CGI.

Fréquentation : Le nombre et le type des visites sur un site.

Frameset : Jeu d'encadrement qui apparaît dans un navigateur et permet de visualiser simultanément plusieurs pages Web.

FTP (*File Transfer Protocol*) : Protocole standard pour le transfert de fichiers en tous sens sur un réseau.

Fonction : En programmation, un fragment de programme renfermant une série d'instructions, pouvant ensuite être appelé ou utilisé à plusieurs reprises dans le programme.

G

Gadget : Objet miniature offrant diverses fonctionnalités. C'est le nom donné par Google à ses widgets.

GIF : Format de fichier graphique bien adapté aux images simples.

Glisser-déposer : Fait de cliquer sur un objet avec le bouton gauche, puis de déplacer la souris en laissant le bouton enfoncé et de déposer ensuite l'objet (en relâchant le bouton) à un autre emplacement de l'écran.

Google : À l'origine un simple mais très performant moteur de recherche, aujourd'hui un groupe qui propose de nombreux produits et solutions aux internautes (Google Maps, Google Search, Google Earth, Google Gadgets, etc…)

Google Analytics : Système de suivi de la fréquentation d'un site, proposé par Google.

GUI (*Graphical User Interface*) : L'interface graphique utilisateur est la partie graphique d'un programme qui, le plus souvent à l'aide de fenêtres et de boîtes de dialogue, permet de contrôler ce programme. Par opposition aux programmes de type texte, à interface ligne de commande, souvent plus difficile à apprendre et à utiliser.

H

Headings : En HTML, les *headings* (en-têtes) constituent un moyen tant visuel que structurel d'organiser et de présenter un document.

HEAD : Élément d'un document qui renferme des informations sur la page, mais pas de contenu visible.

Hit : Terme anglophone qui correspond à une visite sur un site.

Hébergement Web : Fait d'avoir recours à un serveur Web qui appartient à une tierce personne pour rendre un site Web disponible sur Internet. Votre hébergeur est généralement votre FAI.

Hôte : Nom d'un ordinateur ou d'un programme sur un domaine, comme le `www` de **www.domain.com** ou `local2` dans **local2.domain.com**

HTML (*Hypertext Markup Language*) : Ensemble des éléments identifiés qui indique à l'agent utilisateur comment afficher ces éléments et les informations d'une page Web.

HTTP (*Hypertext Transfer Protocol*) : Protocole employé par les ordinateurs lors du transfert de documents Web (`http://`).

I

Icône : Petite image aidant à la navigation ou apportant une signification accrue au texte.

ID : Sélecteur de feuilles de style devant être unique à l'intérieur d'une page Web, et pouvant utiliser n'importe quel style.

Image cliquable : Une même image comportant plusieurs liens hypertextes.

Imbriquer : Placer un élément HTML à l'intérieur d'un autre élément. Fréquemment utilisé pour des éléments identiques, tels que listes et tableaux. Certains éléments ne peuvent être imbriqués.

Indice : Texte placé en dessous de la ligne normale de texte.

Input : Accepter des données à partir d'un site Web, en général *via* un champ de formulaire.

Interactivité : Accepter des saisies utilisateur et y répondre de façon dynamique.

Internationalisation : Rendre un site accessible à d'autres langues et cultures. Également nommé localisation.

Internet : Réseau international et publiquement disponible d'ordinateurs utilisant des protocoles standard réseau pour communiquer et transférer des informations.

Internet Explorer : Le navigateur Web Microsoft intégré aux systèmes d'exploitation Windows.

Intranet : Réseau privé interne, souvent placé à l'abri d'un pare-feu.

ISP (*Internet Service Provider*) : Fournisseur d'accès Internet (FAI) : la société ou l'organisation qui procure l'accès Internet à un individu ou à une société.

J

Java : Langage de programmation complet et compatible entre plates-formes développé par Sun Microsystems.

JavaScript : Langage de scripts côté client développé par Netscape.

Journal : Fichier d'enregistrement d'événements précis. Par exemple, chaque accès à un site Web : les fichiers concernés et d'autres informations telles que l'heure, la date et l'adresse IP de chaque client.

JPEG : Format de fichier graphique particulièrement adapté aux images complexes (photographies).

JScript : Implémentation Microsoft de JavaScript : JScript et JavaScript ne sont pas 100 % compatible entre eux.

L

Libre (source) : Tout programme technique ou bibliothèque dont le code source est librement disponible sous forme non compilée, généralement afin de permettre le développement par des programmeurs individuels.

Lien hypertexte : Mot, expression ou image utilisant une marque HTML qui en fait un lien « cliquable ». En cliquant sur un lien hypertexte, l'utilisateur voit son navigateur Web ouvrir la ressource correspondant au lien.

Lien (vérification de) : Utilisation d'un outil Web pour vérifier tous les liens d'une page Web et s'assurer qu'ils sont valides.

Ligne : Données affichées horizontalement dans un tableau.

Ligne (élément) : Élément n'entraînant pas de saut de ligne ni de rupture de paragraphe lors de sa fermeture.

Localisation : Rendre le site accessible à une langue ou une culture particulière.

M

Masqué (champ) : Champ de formulaire non visible par l'utilisateur, mais néanmoins transmis avec les autres champs de données.

mailto: : Protocole à utiliser à la place de `http://` pour créer un lien de messagerie.

Marge Espace autour de l'extérieur d'un élément.

META : Élément HTML procurant des informations complémentaires telles qu'en-têtes HTTP, date de rafraîchissement ou d'expiration de fichiers, noms des auteurs, mots-clés et tout autre contenu possible.

Méthode (de formulaire) : Façon dont les données du formulaire sont transmises au serveur : `GET` ou `PUT`.

Modèle : Document préconçu dans lequel il suffit de « remplir les blancs » pour obtenir un document utile et agréable.

Module complémentaire : Application tierce installée dans une autre application comme programme intégré, tels Shockwave ou Acrobat Reader. Également nommé *plug-in*.

Mot de passe : Mot, expression ou chaîne de caractères secrets permettant de vérifier que vous êtes celui que vous déclarez être.

Moteur de recherche : Site Web ou programme dressant le catalogue et indexant les sites Web, puis autorisant les utilisateurs à rechercher un site particulier à l'aide d'un mot-clé, d'une description ou d'autres critères.

Mozilla : Précurseur de Netscape Navigator, et maintenant un mouvement pour les navigateurs à source libre, qui a notamment créé Firefox et Thunderbird.

Multimédia : Combinaison quelconque de textes, graphismes, fichiers audio ou vidéo et pages de réalité virtuelle.

N

Navigateur Web : Programme informatique utilisé par un internaute pour visualiser une page Web.

Navigation : Outils, visuels ou autre, que procure un site Web pour que les utilisateurs puissent se déplacer sur le site.

Netscape Communicator : Ensemble de navigation Web développé et distribué par Netscape Corporation.

name : En HTML, attribut optionnel qui permet normalement à n'importe quel élément de figurer dans un script.

Nom d'utilisateur : Nom qui permet d'identifier un compte utilisateur en ligne. Une adresse électronique revêt souvent la forme `nomutilisateur@nomhote.domaine.com`.

Nom de domaine : Identifiant unique sur Internet permettant la connexion à un ordinateur individuel.

Nom réservé : En programmation, un mot qui ne peut être employé comme variable ou nom de fonction parce qu'il possède déjà une signification particulière dans le langage de programmation.

Non ordonnée (liste) : Liste à puces.

Non proportionnelle (police) : Police de caractères conservant un espacement constant quelle que soit la largeur réelle de la lettre : `ceci est une police non proportionnelle.`

Non standard (navigateur Web) : Navigateur Web ne respectant pas HTML 4, ou par extension affichant une page Web d'une façon sensiblement différente des navigateurs les plus courants, Firefox et Microsoft Internet Explorer.

Note de fin : Note ou commentaire placé à la fin d'un document.

Note de pied de page : Note ou commentaire placé à la fin d'une page Web.

O

Objet : Élément énuméré dans le Modèle Objet de Document (DOM), utilisé ou manipulé par un script côté client.

Objet incorporé : Tout fichier (le plus souvent multimédia) utilisant l'élément `OBJECT` pour être placé dans un document Web.

Open source : Voir **source libre**.

Ordonnée (liste) : Liste numérotée.

P

Page Web : Page individuelle rédigée en HTML.

Palette : Ensemble des couleurs disponibles dans tout programme d'édition graphique.

Pare-feu : Combinaison de matériel et de logiciels placée entre deux réseaux (par exemple, un réseau interne et Internet) afin de limiter les transactions à ce qui est autorisé.

Parent (élément) : Élément renfermant ou contrôlant (dans un script) un autre élément.

Passage à la ligne : Ce qui se passe lorsque l'extrémité d'un texte atteint la limite droite de l'espace de visualisation, dans une fenêtre ou une zone de saisie.

Perl : Langage de programmation non compilé aussi simple que répandu, utilisé pour les scripts CGI.

Pixel : Plus petit élément adressable d'un écran.

Plate-forme : Combinaison de l'architecture (matériel) et du système d'exploitation (logiciel) rendant un environnement informatique unique (Windows NT sur Intel et Windows NT sur DECAlpha, par exemple).

Plug-in : Voir **complément.**

Police : Aspect visuel des caractères d'un texte.

Positionnement : La façon de placer à l'écran des objets par rapport au coin supérieur gauche de l'écran (positionnement absolu) ou par rapport à d'autres objets (positionnement relatif).

Préférences : Dans un navigateur Web, le menu utilisé pour définir les couleurs de police, la page de démarrage, les identités de messagerie et de groupes de discussion, les dispositifs de sécurité et autres options.

Présentation : Positionnement visuel des paragraphes de texte, d'images et d'autres éléments d'un document.

Propriétaire : Technique ou programme au développement très fermé. Une licence doit être acquise par tout utilisateur potentiel d'une telle technique propriétaire. Ces techniques ne sont validées par aucun comité de standardisation.

Proxy (serveur) : Serveur effectuant la connexion à votre place, puis renvoyant le résultat. Souvent utilisé dans des coupe-feu et pour procurer des services de cache à de grands réseaux.

R

Remplissage : Espace entre la bordure d'un élément (s'il en possède une) et le contenu de l'élément (texte, image ou fichier incorporé).

Réseau : Ensemble d'ordinateurs reliés à l'aide de protocoles réseau identiques, tels que TCP/IP (pour Internet).

Résolution : Nombre de pixels, exprimé en largeur x hauteur, pouvant être affichés par un écran.

RGB (valeurs) : Valeurs respectives de rouge, vert et bleu définissant une couleur en HTML.

S

Sans serif (police) : Polices comme `Arial` et `Helvetica`, dépourvues de serifs, les petites lignes en bas des bras et des jambes des lettres (le pied d'un « l », par exemple).

Script : Petit programme pouvant effectuer un nombre illimité d'instructions.

Scripting : Application d'un script à un élément HTML dans un script côté client tel que JavaScript.

Sécurisé (serveur) : Serveur Web utilisant SSL (*Secure Socket Layer*) ou une autre méthode de codage pour protéger tous les échanges de données entre lui et ses clients.

Sécurité (zones de) : Dans Microsoft Internet Explorer, une zone de sécurité est une région, comme un réseau local, un intranet ou Internet, à laquelle vous appliquez des consignes de sécurité particulières.

Serif (police) : Polices comme `Times New Roman` et `Times`, dotées de petites lignes au bout des jambages des lettres (comme le pied d'un « l », par exemple).

Site Web : Ensemble de pages Web reliées par des liens hypertextes et généralement maintenu par un individu ou un groupe.

Surveiller : Conserver une veille constante sur la disponibilité, le temps de réponse et le trafic d'un site Web.

Serveur : Dans une transaction réseau, le serveur gère de nombreuses connexions émanant de nombreux ordinateurs différents, pour transférer les informations d'un endroit à un autre.

SGML (*Standard Generalized Markup Language*) : Méthode de description de langages à balises devant procurer une information visuelle, structurale et syntaxique du contenu d'un document. Ne s'applique pas forcément au Web, ni même à un document électronique.

Signet : Avec Netscape et Firefox, un site fréquemment visité dont vous enregistrez l'URL dans un fichier sur votre disque. L'équivalent d'un Favori Internet Explorer.

Signet interne : En HTML, un lien hypertexte menant vers une destination interne à la page.

Sortie (output) : Données ou résultat renvoyé après qu'un utilisateur a demandé une information à un serveur Web. Se réfère généralement à une requête.

Source : Voir *code*.

Spécifique navigateur : Élément et/ou fonctionnalité ne fonctionnant que sur une famille de navigateurs et n'appartenant pas aux standards Internet.

Spam : Pourriel**.** Message électronique non sollicité, en général de nature commerciale, envoyé à des gens n'ayant rien demandé et ne souhaitant pas les recevoir.

Spécification : Description de la mise en œuvre d'une technique. La spécification HTML décrit la mise en œuvre préconisée de HTML dans un navigateur Web afin que les auteurs Web puissent s'en servir efficacement.

SSI (*Server-Side Includes*) : Méthode côté serveur qui permet l'inclusion de données, fichiers et informations dans une page Web avant leur affichage dans le navigateur Web.

SSL (*Secure Sockets Layer*) : Standard de sécurité pour les communications TCP/IP, le plus souvent employé avec HTTP.

Système d'exploitation : Logiciel fournissant tous les services de base, tel l'accès aux ressources matérielles, à la mémoire, au processeur et aux autres programmes.

Style (propriété) : Caractéristique individuelle d'un élément HTML modifié ou défini à l'aide d'une feuille de style externe ou interne, ou de l'attribut `style`.

Style (sélecteur) : Attribut HTML (`ID` ou `class`) appliquant des propriétés de style à l'élément contenant l'attribut sans changer l'élément.

Submit (bouton) : Champ de formulaire envoyant les données du formulaire vers un script.

Syntaxe : Façon particulière de rédiger le code source afin qu'il soit compréhensible par un ordinateur.

T

Tableau : Quadrillage permettant d'afficher des données plus efficacement en lignes et en colonnes. Également utilisé dans un simple but de présentation.

TCP/IP (*Transmission Control Protocol/Internet Protocol*) : Utilisés conjointement, ils constituent la forme primaire de transmission de données sur Internet. IP définit le format des paquets de données envoyés tandis que TCP gère le rassemblement des paquets à l'arrivée et la récupération des données perdues.

Téléchargement (download) : Transfert des données depuis un ordinateur distant sur votre propre ordinateur.

Téléchargement (upload) : Transfert de données ou de fichiers de votre ordinateur local vers un ordinateur distant.

Texte (éditeur de type) : Éditeur demandant la saisie manuelle de tous les codes HTML.

Texte seul (navigateur) : Navigateur Web qui n'affiche que du texte : pas d'image, de contenu multimédia ni même souvent de tableaux.

Thunderbird : Le nouveau logiciel de messagerie *open source* de Mozilla.

Touche (hit) : Accès vers une page Web. Correspond en général à un accès vers la page elle-même, pas vers les images qu'elle contient.

Trafic : Portion de bande passante utilisée par le réseau à un moment précis. Un trafic important sur votre serveur signifie de nombreuses touches, mais également des connexions ralenties.

TWAIN (compatible) : Scanner, caméra, appareil photo ou tout autre dispositif d'entrée compatible avec le standard TWAIN.

U

Unicode : Le système de codification de caractères permettant l'emploi électronique de toutes les langues mondiales.

URI (*Uniform Resource Identifier*) : Connu auparavant sous le nom d'URL, c'est l'emplacement d'un document ou d'un fichier sur Internet.

URL (*Uniform Resource Locator*) : Adresse d'identification d'une ressource Internet (le plus souvent une page Web).

Url (codification) : Méthode de codification de caractères spéciaux telles les barres obliques (slashes) dans une adresse URL afin qu'un serveur, ou un autre programme, puisse les comprendre.

Utilisateur : Individu accédant à un site Web ou utilisant un programme.

Uuencode : Connu initialement sous le nom de *UNIX-to-UNIX Copy*, c'est une façon de transformer un fichier binaire en texte dans le but de le transférer à l'aide d'un programme texte seul, tel un client de messagerie électronique.

V

Valeur : Dans un élément HTML, les attributs possèdent le plus souvent une valeur qui modifie le comportement de l'élément.

Validation : Exécuter un programme pour s'assurer que le code HTML utilisé dans une page Web correspond bien au standard HTML 4 du W3C.

Valide : Terme appliqué à un document qui respecte scrupuleusement la DTD à laquelle il fait référence (XML principalement).

Variable : Dans un programme ou un script, un emplacement nommé où des données peuvent être stockées et récupérées selon les besoins des programmes.

VBScript : Un langage de script conçu par Microsoft similaire par sa syntaxe à Visual Basic.

Vignette : Image réduite en taille, qui accélère le téléchargement. Utilisé pour les catalogues d'images et les aperçus. Également nommée « miniature ».

W

W3C (*World Wide Web Consortium*) : le groupe de travail sur les standards à l'origine des recommandations Web tels HTML, CSS et XML.

WebTV : Technique en développement permettant aux utilisateurs de surfer sur Internet à l'aide d'une télévision plutôt que d'un ordinateur.

Widget : Contraction de *windows gadget* correspondant à un dispositif graphique affiché à l'écran. Ce terme est désormais réservé à un objet miniature, généralement programmé en JavaScript, qui accomplit des fonctions particulières.

World Wide Web ou WWW : Ensemble des ressources Web présentes sur Internet.

WYSIWYG (*What You See Is What You Get*) : Tel écran, tel écrit : éditeur dans lequel vous éditez un document tout en pouvant l'examiner sous l'aspect exact qu'il aurait sous un navigateur.

X

XML (*eXtensible Markup Language*) : Sur-ensemble de HTML autorisant l'application de différentes définitions de types de document (DTD) à une page Web.

XSL : Feuilles de style en XML.

Chapitre 14

Index

!

!DOCTYPE 47
1st Page 2000 31

A

A Smaller Gif 208
A, élément 59, 492
Abbr, attribut 441
ABBR, élément 151, 492
Accept, attribut 341
Accept-charset, attribut 340
Accessibilité 438
Accesskey, attribut 366
ACRONYM, élément 151, 492
Action, attribut 340
ADDRESS, élément 147, 492
Adobe Acrobat Reader 455
Adobe Golive 29
Agent utilisateur 42
AJAX (Asynchronous Javascript And XML) 412
Align, attribut 66, 94, 102
Aligner
  cellule 102
  paragraphe de texte 67
  tableau 94
Alink, attribut 177
Alt, attribut 196, 440
Amaya 32, 82
Animated Shop 3 208
Appareil photo numérique 235
Applet 424, 492
APPLET, élément 155, 426
AREA, élément 210, 492
Attribut 23
  abbr 441
  accept 341
  accept-charset 340
  accesskey 366
  action 340
  align 66, 94
  alt 196, 440
  axis 452
  bgcolor 184

char ... 132
charoff ... 132
charset ... 465
class ... 140
codebase ... 226
color ... 166, 177
cols ... 247
content ... 50
data ... 226
de l'élément IMG ... 224
de l'élément OBJECT ... 233, 426
de l'élément FRAME ... 258
dir ... 461, 464
disabled ... 368
enctype ... 340, 374
face ... 166
frameborder ... 258
headers ... 441
height ... 194
href ... 213, 305
HTTP-EQUIV ... 419
id ... 62, 140, 269, 287
ismap ... 222
lang ... 140, 145, 457
link, vlink et alink ... 177
longdesc ... 259
marginwidth et marginheight ... 258
media ... 291, 305
method ... 340, 370
name ... 50, 62, 257, 340
nohref ... 220
noresize ... 258
readonly ... 369
rel ... 304
rows ... 247
scope ... 441
scrolling ... 258
shape ... 214
size ... 165
span ... 124
src ... 257
style ... 148, 281
summary ... 93, 128, 439
tabindex ... 363
target ... 261
text ... 177

title 196, 304
type 74, 76, 226, 287, 305, 343, 386
usemap 231
width 104, 124, 194
Axis, attribut 452

B

B, élément 164, 493
Background, attribut 187
Balise 22
BASE, élément 264, 493
BASEFONT, élément 165, 493
BDO, élément 463, 493
Bgcolor, attribut 184
BIG, élément 164, 493
Bloc-notes 34
BLOCKQUOTE, élément 143, 493
BODY, élément 53, 493
Border, attribut 95
BR, élément 65, 493
BUTTON, élément 350, 494

C

Calculatrice 182
CAPTION, élément 93, 494
Caractère
encodage de 157
langues et jeux de 465
non affichable 163
spéciaux 154, 159
Casse, sensibilité à la 24
Cellpadding, attribut 99
Cellspacing, attribut 99
CENTER 494
CGI (Common Gateway Interface) 415
Char, attribut 132
Charoff, attribut 132
Charset, attribut 465
Citation
créer 143
imbriquer 145
Cite, attribut 143

CITE, élément 143, 155, 494
Class, attribut 140
CNIL (Commission Nationale Informatique et Liberté) 375
Code source
  afficher 52
CODE, élément 152, 494
Codebase, attribut 226
COL, élément 122, 494
COLGROUP, élément 122, 494
Color, attribut 166, 177
Cols, attribut 247
Colspan, attribut 107
Commentaire
  HTML 137
  JavaScript 390
  SGML 389
  VBScript 390
Composer 33
Compte
  Google Analytics 478
Content, attribut 50
Cookie 416
Corell Draw 180
Couleur
  ajouter à une page Web 177
  modifier dans un élément 183
  modifier dans un tableau 184
  modifier le schéma de 178
  modifier pour une cellule de tableau 184
  noms et valeurs RGB 177
  précautions à prendre 184
Créer
  bloc de texte comme pré-formatté 141
  citation 143
  commentaire HTML 137
  couleurs 177
  GIF animé 199
  image 190
  image cliquable 214
  image d'arrière-plan 187
  jeu d'encadrement 245
  lien hypertexte 59
  ligne horizontale 136
  liste à puce 75
  liste de définitions 76
  liste ordonnée 73

listes imbriquées 79
notation mathématique 169
page d'accueil 47
paragraphe de texte 57
saut de ligne 65
tableau 89
tableau à cellules fusionnées 107
titre 54
CSS (Cascading Style Sheet) 18, 280
propriétés de boîtes 311
propriétés de texte et de police 310
sélecteur 282
CutePage (voir SoThink) 36

D

Data, attribut 226
DD, élément 76, 495
DEL 495
DFN, élément 155, 495
DHTML (Dynamic HyperText Markup Language) 19
DIR 495
Dir, attribut 461, 464
Disabled, attribut 368
DIV, élément 140, 288, 495
DL, élement 76, 495
DOCTYPE, élément 245
DOM (Document Object Model) 395
DOM2 (Document Object Model level 2) 20
Donnée (partager entre des cadres) 269
Dpi (dots per inch) 234
Dreamweaver 29
DT, élément 76, 495
DTD (Document Type Definition) 16

E

ECMAScript 19, 412
Editeur 28
1st Page 31
Adobe Golive 29
Amaya 32
Composer Mozilla 33
Dreamweaver MX 2004 29

FrontPage Express ... 33
FrontPage) ... 29
HotDog ... 30
Namo WebEditor ... 30
NotePad ... 34
Nvu ... 35
SoThink ... 36
Web Fast ... 31
WebExpert ... 30
Word ... 31
WYSIWYG (What You See Is What You Get) ... 28
Elément ... 22
A ... 59
ABBR ... 151
ACRONYM ... 151
ADDRESS ... 147
APPLET ... 155, 426
AREA ... 210
B ... 164
BASE ... 264
BASEFONT ... 165
BDO ... 463
BIG ... 164
BLOCKQUOTE ... 143
BODY ... 53
BR ... 65
BUTTON ... 350
CITE ... 143, 155
CODE ... 152
de type bloc ... 23
de type ligne ... 23
DFN ... 155
DIV ... 140, 288
DL ... 76
DOCTYPE ... 47, 245
EM ... 149
EMBED ... 430
FIELDSET et LEGEND ... 360
FONT ... 165, 183
FORM ... 339
FRAME ... 248
FRAMESET ... 245
HEAD ... 49
Hn ... 54
HR ... 136
HTML ... 48

I 164
IFRAME 271
IMG 155, 191
INPUT 343
ISINDEX 350
KBD 152
LABEL 357
LINK 303
MAP 210
META 50, 157, 280, 305, 386
NOFRAMES 246, 270, 453
NOSCRIPT 388
OBJECT 155, 224, 269, 426, 455
OL 73
P 57
PARAM 429
parent 23
PRE 141
Q 143
S 164
SCRIPT 383
SELECT 352
SMALL 164
SMP 152
SPAN 140, 288
STRIKE 164
STRONG 149
structure 24
STYLE 282
SUB 167
SUP 167
TABLE 90
TD 90
TEXTAREA 356
TITLE 49
TR 90
TT 164
U 164
UL 75
VAR 155
EM, élément 149, 495
EMBED, élément 430
Encodage de caractères 157
Enctype, attribut 340, 374
Entités de caractères 160
Evénements 390

F

Face, attribut ... 166
Feuille de styles
alternative ... 304
auditive ... 328
en cascade ... 323
externe, créer ... 294
héritage ... 323
interne, créer ... 282
principales propriétés CSS ... 309
propriétés de boîtes ... 311
FIELDSET, élément ... 360, 495
Filezilla ... 475
Firefox ... 39
FONT, élément ... 165, 183, 495
For, attribut ... 358
FORM, élément ... 339, 495
Format HTML ... 49
Formulaire
ajout d'une structure ... 360
bouton d'action ... 348
bouton d'option ... 346
bouton de soumission graphique ... 347
bouton de soumission ou de réinitialisation ... 346
boutons ... 350
case à cocher ... 345
commande cachée ... 348
commandes en lecture seule ... 369
commandes inactives ... 368
et focus ... 362
menu ... 352
saisie de mot de passe ... 344
saisie de texte ... 343
soumettre ... 370
touches d'accès rapide ... 366
traitement des données ... 373
traiter à l'aide d'un script ... 415
zone de texte de saisie multilignes ... 356
Frame (voir Jeu d'encadrement) ... 244
Frame, attribut ... 95
FRAME, élément ... 248, 496
Frameborder, attribut ... 258
Frameset (voir Jeu d'encadrement) ... 244
FRAMESET, élément ... 245, 496
Fréquentation, suivre la ... 478

FrontPage 29
FrontPage Express 33
FTP (File Transfer Protocole) 474

G

Get, méthode HTTP 371
Gif gIf giF 208
Google Analytics 478
compte 478

H

HEAD, élément 49, 496
Headers, attribut 441
Height, attribut 194
Héritage
des attributs 130
feuille de styles 323
Hn, élément 54, 496
HotDog 30
HR, élément 136, 496
Href, attribut 213, 305
HTML 496
commentaire 137
définition 12
éditeur 28
élément 48
éléments structuraux de texte 155
événements intrinsèques 391
format 49
HTTP (HyperText Transfer Communication Protocol) 60
HTTP-EQUIV, attribut 419

I

I, élément 164, 496
Id, attribut 62, 140, 269, 287
IETF (Internet Engineering Task Force) 12
IFRAME, élément 271, 497
Image
animée 198
d'arrière-plan, inclure 187

insérer 190
réduire la taille des fichiers 237
réduire la taille physique 234
IMG, élément 155, 191, 497
Index.html 247
INPUT, élément 343, 497
INS 498
Instruction
!DOCTYPE 47
document.write 384
Internationalisation 456
Internet Explorer 39
ISINDEX, élément 350, 498
Ismap, attribut 222

J

Java (différences avec Java script) 425
JavaScript 397
compatibilité avec les navigateurs 407
différences avec Java 425
propriétés de document 406
Jeu d'encadrement 244
accessibilité 453
cadre cible 261
cible par défaut des liens 264
conflits de noms 263
contenu de remplacement 270
imbriquer 265
noms réservés 262
partage de données 269
JVM (Java Virtual Machine) 455

K

KBD, élément 152, 498

L

LABEL, élément 357, 498
Lang, attribut 140, 145, 457
valeurs 459
LEGEND, élément 360, 498
Légende de tableau 93

LI, élément 73, 498
Lien hypertexte, créer 59
Link, attribut 177
LINK, élément 303, 498
Liste
  à puce, créer 75
  de définitions, créer 76
  imbriquée, créer 79
  ordonnée, créer 73
Listing
  accueil.html v. 2.7.1 252
  audit.css 333
  barrenav.html 2.7.1 251
  citations imbriquées 146
  date2.js 402
  élément CODE 151, 153
  élément MAP restitué 230
  éléments ABBR et ACRONYM 151
  éléments OBJECT et EMBED 430
  exemple MathML 172
  extrait de la section HEAD de la page d'accueil de MicroApplication 53
  extrait du code du fichier listesimbriquées2.html 87
  famille.html (version 1.4.1) 92
  fonction Get_Cookie 420
  index.html v.2.7.1 250
  index.html v.2.7.2 256
  listes imbriquées 80
  notations scientifiques en texte 169
  pageacc1_0.html. 57
  pageacc1_3_2.html 64
  pageacc1_3_3.html 68
  Pageacc_v1_3_3.html 46
  position.html 318
  position0.html 316
  region.html version 1.4.2 110
  region.html version 3.8.1 298
  region_erreurs.html 120
  romeo.html 329
  styleaudit.css 329
  styleaudit2.css 332
  STYLEDIVSPAN.html 289
  tailles de police 166
  visuel.css 333
Localisation 456
Longdesc, attribut 259
Lynx 41

M

MAP, élément 210, 499
Marginheight, attribut 258
Marginwidth, attribut 258
Masquer (les données de script aux agents utilisateurs) 389
Mathématiques, afficher en HTML 169
MathML 171
Media, attribut 291, 305
@media, règle 305, 332
MENU 499
META, élément 50, 157, 280, 305, 386, 499
Method, attribut 340
Microsoft Gif Animator 204
Modèle MVC (Model-View-Controller) 279
Modifier
  apparence du texte 164
  couleur d'une cellule de tableau 184
  couleur du texte dans un élément 183
  couleurs d'un tableau 184
  marges d'un paragraphe 148
  police 165
  schéma de couleurs de la page 178
  site pour l'adapter à plusieurs langues 456
  styles affichés par le navigateur 325
Module complémentaire 227, 454
Mosaic 12
Mozilla 39
Multiple, attribut 353
MVC (Model-View-Controller), modèle 279

N

Name, attribut 50, 62, 257, 340
Namo WebEditor 30
Navigateur 39
  activer ou désactiver les scripts 384
  afficher le code source 52
  compatibilité JavaScript 407
  Internet Explorer 39
  Lynx 41
  modifier les styles affichés 325
  Mosaic 12
  Mozilla/Firefox 39
  Opera 39

Safari 39
Navigator, objet 410
NOFRAMES, élément 246, 270, 453, 499
Nohref, attribut 220
Noresize, attribut 258
NOSCRIPT, élément 388, 499
NotePad (voir Bloc-notes) 34
Numéro de version 63
Nvu 35

O

OBJECT, élément 155, 224, 269, 426, 455, 499
Objet navigator 410
OL, élément 73, 499
Opera 39
OPTGROUP, élément 354, 500
OPTION, élément 353, 500

P

P, élément 57, 500
Page d’accueil 47
Paint 180
Paint Shop Pro 180, 239
Paragraphe
aligner 67
créer 57
modifier les marges 148
PARAM, élément 429, 500
PhotoShop 180
Plug-ins (voir Module complémentaire) 156
Popup (voir Fenêtre surgissante) 404
Post, méthode HTTP 371
PRE, élément 141, 500
Précaution avec les couleurs 184
Propriétés CSS 310
PwWebSpeack 467
PwWebspeak 453

Q

Q, élément 143, 500

R

Readonly, attribut 369
Référence de caractère 154
entité 160
numérique 159
Règle @media 305, 332
Rel, attribut 304
Résolution 236
Rows, attribut 247
Rowspan, attribut 107
Rules, attribut 95

S

S, élément 164, 500
Safari 39
SAMP 500
Saut de ligne, créer 65
Scope, attribut 441
SCRIPT 500
à action unique 381
activer ou désactiver dans le navigateur 384
CGI (Common Gateway Interface) 415
cookie 416
côté client 380
côté serveur 380
ECMAScript 412
événements intrinsèques HTML 390
JavaScript 397
masquer les données aux agents utilisateurs 389
répétitif 382
traitement de formulaire 415
VBScript 413
SCRIPT, élément 383, 500
Scrolling, attribut 258
SELECT, élément 352, 500
Sélecteur CSS 282
Sensibilité à la casse
HTML 24
XHTML 24
SGML (Standard Generalized Markup Language) 12
Shape, attribut 214
Shockwave 455

Site Web
hébergement ... 473
publier sur le Web ... 472
suivre la fréquentation ... 478
test ... 466
Size, attribut ... 165
SMALL, élément ... 164, 501
SmartSaver 3.0 ... 239
SMP, élément ... 152
SoThink HTML Editor ... 36
Span, attribut ... 124
SPAN, élément ... 140, 288, 501
Src, attribut ... 191, 257
Start, attribut ... 74
STRIKE, élément ... 164, 501
STRONG, élément ... 149, 501
Style, attribut ... 148, 281
STYLE, élément ... 282, 501
masquer le contenu ... 292
SUB, élément ... 167, 501
Summary, attribut ... 93, 128, 439
SUP, élément ... 167, 501

T

Tabindex, attribut ... 363
TABLE, élément ... 90, 501
Tableau
accessibilité ... 441
aligner ... 94
aligner une cellule ... 102
attributs de bordure et de règles de l'élément TABLE ... 96
bordures et règles, attributs border, frame et rules ... 95
calcul du nombre de colonnes ... 129
cellules manquantes et cellules se recoupant ... 117
colonne, élément TD ... 90
créer ... 89
en-tête de colonne ... 93
fixer la largeur des colonnes ... 104
fusionner des cellules ... 107
héritage des attributs ... 130
imbriquer ... 127
légende de tableau, élément CAPTION ... 93
ligne, élément TR ... 90
mise en forme, attributs char et charoff ... 132

modifier la couleur d'une cellule 184
modifier le schéma de couleurs 184
regrouper des colonnes 122
regrouper des lignes 121
taille et structure des cellules 98
Tabulation (ordre de) 363
Target, attribut 261
TBODY, élément 121, 501
TD, élément 90, 502
Test du site Web 466
Text, attribut 177
TEXTAREA, élément 356, 502
Texte
éléments structuraux 155
modifier l'apparence 164
modifier la couleur du texte dans un élément 183
structurer 149
TFOOT, élément 121, 502
TH, élément 93, 503
The Gimp 180, 216
THEAD, élément 121, 503
Tim Berners-Lee 12
Title, attribut 196, 304
TITLE, élément 49, 503
Titre, créer 54
TR, élément 90, 503
TT, élément 164, 503
Type, attribut 74, 76, 226, 287, 305, 343, 386

U

U, élément 164, 503
UL 504
Ulead Gif Animator 204
UNICODE 461
Urchin.js, bibliothèque JavaScript 479
UrchinTracker(), fonction JavaScript 479
URI (Universal Resource Identifier) 59
URL (Universal Resource Locator) 59
Usemap, attribut 231

V

Valign, attribut 102
Value, attribut 74
VAR, élément 155, 504
VBScript 413
Version, numéro de 63
Vlink, attribut 177

W

W3C (World Wide Web Consortium) 12
Web Fast 31
WebExpert 30
WebFTP 474
Width, attribut 104, 124, 194
Word 31
WS_FTP95 LE 476

X

XHTML (eXtensible HyperText Markup Language) 16, 21
XML (eXtensible Markup Language) 17

Composé en France par Jouve
11, bd de Sébastopol - 75001 Paris